CROSS

A DATA LOGGING AND TELEMETRY SYSTEM

Principal Investigator

David G. Harkrider

Prepared by

Robert F. Nickerson

Seismological Laboratory
California Institute of Technology
Pasadena, California 91125

November, 1979

FINAL REPORT

5 January 1977 - 30 September 1979

Contract Number:	U.S. Geological Survey Contract No. 14-08-0001-16629
Name of Contractor:	Seismological Laboratory California Institute of Technology Pasadena, California 91125
Principal Investigator:	David G. Harkrider Seismological Laboratory, 252-21 California Institute of Technology Pasadena, California 91125
Government Technical Officer:	Gordon W. Greene U.S. Geological Survey OES External Research Program, MS 83 345 Middlefield Road Menlo Park, California 94025
Short Title of Contract:	Establishment of a Southern California Geophysical Data and Analysis Center
Effective Date of Contract:	January 5, 1977
Contract Expiration Date:	September 30, 1979
Amount of Contract:	$375,068
Date of Report:	November 1, 1979

Sponsored by the
U.S. Geological Survey
Contract No. 14-08-0001-16629

Table of Contents

CROSS - A DATA LOGGING AND TELEMETRY SYSTEM

1 FOREWORD

We are in the stage in earthquake prediction research where it is necessary to provide the scientific community with immediate access to a wide variety of geophysical data. The Palmdale uplift and other geodetic and seismic-velocity data have focused attention on Southern California.

Progress in the field of earthquake prediction can be substantially accelerated if widely separated researchers have simple and immediate access to a broad spectrum of remotely sensed data. Not only would this stimulate advances in the field of eathquake prediction, but also instrumental redundancy at various research institutions could be substantially reduced. This would in turn allow a more effective allocation of available funds to the core problems of earthquake prediction. The tantalizing but limited success of studies based on limited sets of data further suggests that a unified data acquisition and storage system could have a profound effect upon the course and direction of earthquake prediction research. Indeed, future progress may well depend on the cross-analysis of several different types of instrumental data. To this end the program to design a data acquisition and exchange network capable of keeping pace with the rapidly expanding needs of the field of earthquake prediction in the years to come was proposed in 1977 and funded early in 1978. Out of this program came two significant data acquisition systems. They are CEDAR (Caltech Earthquake Detection and Recording system) and CROSS (Caltech Remote Observatory Support System). This report will only deal with the latter. CEDAR is treated in a separate report.

2 INTRODUCTION

Before discussing the details of design it is appropriate to discuss the kind of data involved. The primary data to be included in the CROSS network data base are time series data obtained from ultralong period instruments such as tiltmeters, gravimeters, creepmeters, strainmeters, etc., which are currently being recorded and archived at various institutions throughout California.

There are several criteria that must be met in the design of such an acquisition network. Probably the most critical criterion is the minimization of the time between acquiring of the data in the field and its appearance in a central data storage facility, particularly with respect to identification and action on short term precursory anomalies. Considering the number of experimental sites, some of them very remote and not conveniently accessible, cost can be a significant factor.

In pursuing these goals we have designed and implemented a remote sampling data acquisition network consisting of two functional parts shown schematically in Figure 1. The parts of this network will be discussed in generality here and in detail in succeeding sections. The first part of the network is concerned with data acquisition, that is, getting the data from the remote sensing site to a central collection point and from there to the data center. The data are collected through the use of an on-site microprocessor system which we refer to as a Telemetry Interface Module (TIM). The TIM performs on-site digitization, storage of data, and transmission of the data over a dial-up telephone line when requested by the Data Center computer. Dial-up telemetry also permits the use of Wide Area Telephone Service (WATS) if the number of sites is significant thus greatly reducing operational costs over conventional systems utilizing dedicated leased lines. The Data Center consists of a "PRIME 750" super minicomputer located at the Caltech Seismological Laboratory.

The problems associated with data acquisition can be grouped into two categories depending upon whether long-period, low data rate or short-period high data rate instrumentation is being discussed. We are only concerned here with the former, e.g. tiltmeters, gravimeters, etc. The telemetry Interface Module (TIM) is a device designed to meet the needs for interfacing such long-period remote sensing devices to a central computer over a voice grade dial-up telephone line. Input to the TIM consists of an arbitrary, slowly varying analog voltage level. The major functions of the TIM are digitization, data compaction if desired, storage, and subsequent digital transmission to a central computer. It may also be neccessary to use some form of signal conditioning between the instrument signal output and the TIM so that the signal applied to the TIM input is compatible with it.

Flexibility is obtained through the use of a 12-bit microprocessor (INTERSIL IM6100). Analog data is sampled, converted to digital and stored on site in random access memory. Low initial cost is attained through the use of standard hardware coupled with the flexibility

inherent in the use of a microprocessor, obviating redesign to accommodate diverse instrumentation. The digital data are retained until they are requested, usually once each day, but any other desired schedule is possible, on a dial-up line. This telemetry technique reduces the operational costs and risks involved compared with options requiring leased dedicated lines or frequent visits by field personnel to physically retrieve data storage media. Additionally, the hardware required at the receiving site or data center is greatly simplified, especially at the data center. This hardware consists of a MODEM with an automatic dial-out feature interfaced to the Data Center computer. Consequently, no hardware modifications whatsoever are required in order to accommodate the addition of stations to the telemetry network. As designed, the TIM can have power failure protection capability and can survive power loss for a period in excess of one month or less depending on the size of the backup battery. In case of loss of communication, sampling will continue until the local memory is full although "wrap around" storage is also possible. The data can also be physically recovered at the site either by the actual connection of a mobile terminal with cassette recorder or by swapping memory units in the field for transportation to the data center at Caltech. As a result, we can maintain data acquisition even during the most catastrophic of circumstances. Maintenance costs are minimized by a modular design allowing most maintenance tasks to be performed at a central site rather than in the field. Further reduction in operating cost is attained by allowing a cluster of as many as eight remote sensing instruments (gravimeters, tiltmeters, etc.) to function through a single TIM. The latter would be particularly useful for establishing telemetry for a remote geophysical observatory.

On-site digitization has the advantage of providing extremely large dynamic range (72 db) while requiring that no concessions be made with respect to sensitivity. Such is certainly not the case for analog telemetry.

Timing (required for complete site isolation) is provided by a crystal oscillator that is resynchronized to the data center system at the end of each data interrogation.

3 PROJECT DEVELOPMENT

The development of the CROSS project occurred in four stages:

1. Goals and specifications for CROSS were established as well as the design criteria of the TIM subsystem hardware. A proof of the concept was done using the Caltech NOVA 1200 computer system along with an Intersil Intercept 6100 prototyping system that demonstrated the digital telemetry data transmission concept, analog-to-digital conversion techniques and the application of the OMEGA transmission for timing all under control of the 6100 microprocessor. A computing system for the Data Center was selected (Eclipse S/230) and ordered.

2. The Eclipse was delivered. Initial development of the CROSS software was accomplished using the Eclipse and the prototyping system. Design goals were changed somewhat by deleting the OMEGA system and substituting a TCXO oscillator as a simpler, more reliable method for local time reference. The concept of gain control for incoming signals was deleted to be replaced by an external signal conditioning box controllable by the TIM. This would be designed and made available only if needed. The design and construction of an operating TIM prototype was initiated.

3. TIM prototype was completed. Final development of the CROSS system software was done using the TIM prototype. Design of a field deployable TIM based on but not identical to the prototype was finished and an order placed for 19 units based on this design was placed.

4. TIM units were received from the vendor and the first few deployed at instrumentation sites. The CROSS system was inaugurated and data was collected on a regular schedule. The system was moved to the PRIME computer.

4 OVERVIEW OF CROSS

As previously stated the CROSS concept was envisioned as an economical way of collecting and retrieving data from remotely sited long period geophysical instrumentation. It relies on the use of the aforementioned TIM that acquires data from a multiple number of nearby instruments, stores this data and transmits the acquired data upon demand over standard direct-dial voice-grade telephone lines (see Figure 1).

Data transmission is initiated by a central computer system located at the Caltech Geophysical Data Center. The central computer is a PRIME "750".

All control of the CROSS system resides in the Data Center computer software. When a set of TIM units is to be interrogated, the program fetches a TIM unit entry from a station list that is normally resident in a disc file. This station list consists of an entry for each TIM site (see Figure 2). Each entry consists of the station ID, a control word, and a telephone number. The program uses the telephone number to initiate the call-up sequence using the auto dialer. When the hardware has established a successful linkup with the target TIM, the "750" commences to make requests of the TIM. These requests and their replies take the form of message packets described in Appendix C. The nature of these requests are determined by the control word referred to above. An example of these requests are: "request operational status", "send a block of program memory", "send a block of data memory", or "use the contents of the current message packet as a program segment to be loaded into the program memory". Figure 3 shows the Data Center program flow of a typical TIM interrogation sequence. Appendix C describes the current available interrogation commands. At the conclusion of the interrogation, the TIM resumes its normal data logging and the data set acquired by the master computer is stored in a file that then becomes part of the CROSS data base. The system then commences to "call up" the next TIM site on the list until the list is exhausted.

5 DESCRIPTION OF THE DATA CENTER

The Data Center contains a number of computing systems but the one that concerns this report is a PRIME "750" super minicomputer. The "750" is a 32 bit high speed multiuser virtual memory system with one megabyte of real memory and a 300 megabyte disc storage system. The "750" interfaces to an auto-dialer modem subsystem via RS-232 communication lines and a communication control box that enables programmed dial-up to a target system at some remote site. The Seismological Laboratory acquired the "750" to meet the computing needs of the research personnel. It is fortunate that it is also well suited to the needs of the CROSS project.

An Eclipse S/230 system was originally acquired for the CROSS project. It was used extensively for the development of the CROSS system. The limitation of data storage, single user only accessibility and need for an operator to initiate the call up sequence were major obstacles to the full implementation of a fully automated unattended data acquisition system. For these reasons the project was moved to the PRIME system when it became available.

The operating system of the "750" (called PRIMOS) has a number of features that meet the needs for a totally unattended data acquisition system. It supports a versatile file structure so that as a data dump from each TIM is acquired, the data set can be stored in a file with a filename generated that is related to the identification character of the TIM site and the origin date and time of the data set (see Appendix D). (This is convenient for retrieval purposes.) The system supports phantom jobs, i.e., programs that run independent of a terminal so the data acquisition software can run continuously as a continuous loop phantom although the job is only active for the time it takes to interrogate the various TIM sites usually done at 3 a.m. daily. Since PRIMOS allows each user to have his own file directory, the CROSS program can function without any possible interference from other users. The large amount of storage available allows a fairly large CROSS data base to be resident and on-line at all times. This large data base capability can be important since the data base is available to be acquired by any user logged into the system including remote users via a computer network linkage (proposed but not yet implemented). Thus many different investigators may have ready access to their data set. Until a more permanent solution to the data archival problem is available, the PRIME system will have to function in the stead of centralized data archives at least insofar as the CROSS project is involved.

6 DESCRIPTION OF TIM SUBSYSTEM HARDWARE

The Telemetry Interface Module (TIM) subsystem of CROSS is a microprocessor based data logging and telemetry unit. Its principal characteristics are a capability to collect and store data and to transmit the data, upon command, over standard voice- grade telephone lines. In addition, its power consumption during the data logging periods is less than 1/2 watt. An important design consideration for both the software and hardware was a capability to recover from a software failure using only the telemetry facilities.

The TIM has been designed to be modular at the functional level utilizing a card cage concept with I/O bus backplane. It was also designed to be rugged. The card modules reside in a card cage that is bolted to the front panel cover that serves to retain the cards within their slots. The front panel in turn is bolted to a heavy gage steel case with a cover that has a carrying handle. Signal input terminals are attached to the side of the case, the terminals are electrically connected to the card cage backplane with a cable and connector. The card cage and front panel may be removed from the case as a unit by disconnecting the cable harness. In this manner an entire TIM unit may be replaced without the need to remove and reconnect the several wires attached to the terminal strip. Since the TIM is not handled once it has been installed, the front panel was designed for minimal function. Besides the usual on-off switch, there is a reset pushbutton that is used for software control and four toggle switches also used for software control. There are a number of display LED lamps that are illuminated by a display pushbutton. The display itself is controlled by a three position rotary switch. A more detailed description is given in Appendix F with photos in Appendix H.

The TIM, as stated above, has been designed to be modular at the functional level. Card modules that currently comprise the TIM are:

1. CPU-program memory card.
2. Status and clock card.
3. Data storage RAM card.
4. Analog input card.
5. Digital input card.
6. Digital output card.
7. MODEM controller card.
8. MODEM card.

In addition there is a test card attached to a control box that can be temporarily installed in a TIM unit for hardware checkout or to aid in program debugging. There is also a DAA attached to the case cover.

The TIM subsystem was designed around the Intersil 6100 microcomputer. The 6100 was chosen because it and its family of supporting IC packages belong to the CMOS family of solid state devices. The significant important attribute of CMOS is low power consumption.

The 6100 is designed to be equivalent to a PDP-8 minicomputer therefore much of the software that has been created over the past years for the PDP-8 would be compatible with a system utilizing the 6100. However, the low power requirement and flexible instruction set were the primary factors influencing the choice of the 6100.

General Description and Function of Card Modules

6.1 CPU - Program Memory

This card contains the 6100 CPU microprocessor, 2048-12 bit words of RAM, and a 2708 EPROM. The EPROM is configured as a 512-12 bit word ROM. The intent of the design was that the telemetry software would reside in the preprogrammed non-volatile EPROM and the data logging program would occupy the RAM. Since any ROM resident program cannot be altered or changed, intentionally or unintentionally, a highly reliable telemetry function is possible. That is to say as long as the hardware is functioning properly, the TIM will always be capable of responding to telephone telemetry requests from the central computer even though other functions may have ceased operating properly, if at all, e.g., after a power outage. On the other hand, since the RAM can be altered at will a data logging program can be remotely loaded at any time utilizing the telemetry function resulting in a very flexible system. Secondly, there are three methods by which a reset and restart can take place. They are: manual intervention (pushbutton), a telephone call (ringing circuit detection), and the auto-restart. Any of these occurrences will force the microprocessor to halt its current operating mode and start executing the ROM resident program. In the case of the restart due to a telephone call, restart is delayed so that the necessary setup is accomplished so that the interrupted function may continue at the end of the telemetry call. An auto-restart will only occur if the main program ceases to function properly. In this case, a special monitoring circuit is allowed to "time out" thereby causing a restart. Should this happen the system goes into a software standby mode until the unit is interrogated by the central computer via the telephone call-up.

Central to the restart function is a status register. The register pinpoints the cause of the restart. In this way the ROM resident program can identify the cause and take the program branch that is appropriate to handle the particular cause such as sending a new program load to the TIM. In addition front panel switches control selected bits of the status register so that they also may be used to control the program flow if desired.

Figures 5a and b are a simplified flow chart showing the general flow of the telemetry software. A feature that is not shown is the "time out" monitor. This feature is implemented in software and has no direct relation to the auto-restart monitoring circuit previously referred to. Should there be no telemetry activity for a predetermined number of seconds as would be the case if there was, e.g., a wrong number call or if a break occurred in the telephone circuits, the TIM will "hang up" the telephone line and resume its function at the point of interruption.

6.2 Status

The status card contains system support logic including timing circuitry, a TCXO clock for on board time keeping, the aforementioned 12 bit input status register and a programmable display register. The input status register, which can be tested by a program, referrred to in (6.1), contains information on the cause of a restart, if any, and the position of any of the front panel switches. The output display register in conbination with the front panel switches can be used to show selected parameters on the display lights. The TCXO and related logic produces one (1) second period pulses that can be used by a program to maintain a system clock.

6.3 Data RAM

The data RAM is a solid state CMOS memory capable of storing 4096 12 bit numbers. As data is collected by the system, it can be stored in this RAM until transmitted to the Data Center by the telemetry function. The card contains a Nicad battery as a backup to save the data if the power to the TIM is lost.

6.4 Analog Input

The analog input card contains an eight (8) channel input multiplexer, differential input amplifier and a 13 bit analog-to-digital converter. Under the program control, the multiplier can be set to any of the eight input channels and the voltage on that channel measured by converting it to an equivalent 12 bit digital number. The number can then be stored by the program in the data RAM.

6.5 Digital Input

The digital input board contains four (4) UARTs and therefore can accept serialized digital data on any of four input channels. The data can be received as ASCII characters or as seven or eight bit data bytes in a pseudo ASCII format. Such data when received can be preprocessed if desired, and stored in the data RAM.

6.6 Digital Output

The digital output card contains one UART and a 16 channel output multiplexer. The type of data sent is under program control and is the same as received by the digital input card, i.e., ASCII formatted. The output channel is preselected by the program prior to the transmission. The principal use of this card is for sending control parameters to the target package, e.g., to change the gain setting of a signal conditioning package.

6.7 MODEM Controller

The MODEM Controller is the interface between the TIM system and the MODEM for control of the telemetry function. It contains logic that allows full data set control of the modem as defined by the RS-232 communications standard. It is by means of this controller that a program running in the computer section can answer an incoming call and communicate with the data center with back and forth data transmission using both hardware and software handshaking techniques. This card also contains a switchable 12 volt power supply needed by the CPU PROM and the MODEM card. These two components require a comparatively large amount of power and since they are only needed during the telephone telemetry phase, they are normally switched off. The 12 volts supply is turned on by a reset and turned off by software control.

6.8 MODEM

This board converts serialized digital data from a UART into a modulated carrier tone for transmission over the telephone line. It also demodulates an incoming carrier to a digital signal to send to the UART for data reception.

6.9 Test Card

The principal function of the test card is for trouble shooting and debugging. Associated with the test card is a test box. The combination substitutes for a more elaborate front panel. When plugged into a TIM it can be used to examine and/or change contents of a program memory register, stop a program at a preselected address, step through a program and display the register, address or program instruction. (See Appendix F23).

In summary, the design of the TIM subsystem has the following features:

1. Low operating costs due to low power needs (less than 0.5 watts), and use of standard voice grade direct dial telephone lines for telemetry.

2. Large dynamic recording range since all data is digitized into 12 bits and stored on site. 12 bit digital data may also be acquired.

3. Flexibility -- need only change the microprocessor program or parameter tables in order to meet new requirements.

4. Data integrity -- should the telephone line become unusable, data collection will continue and the data memory module can be returned to the data center for dumping when reachable by field personnel.

5. Program integrity -- programs may be dumped and/or loaded into the TIM using the telephone telemetry capabilities. Telemetry accessibility to the system does not depend on the operating data logging program. Telemetry transmission and reception is controlled by a separate semi-permanently programmed memory subunit (EPROM) that automatically responds to an incoming telephone call. For this reason the telemetry function is essentially failsafe if the hardware continues to function properly.

6. Programmed control of an external signal conditioning unit if needed.

7. Serviceability -- system is modular and individual card modules may be removed and replaced in the field.

8. Rapid access to data since all data collected by the TIM unit is immediately available with a single telephone call to the unit.

9. Adaptability -- e.g., unit could be used as an experiment control system.

10. Low cost of system expansion. Additional TIM units may be added to the network with no changes in existing hardware.

7 DESCRIPTION OF SOFTWARE SYSTEM

7.1 Overview

The CROSS-TIM system is a central master-computer remote slave-computer system connected via standard dial-up telephone lines. Although the remote slave computers form an array of data logging units, this does not constitute a true multislave network since only one slave can be accessed at a time.

The concept of the CROSS-TIM system was such that the following features would be inherent in the design:

1. Reliable data transfer in either direction.

2. A failsafe system such that so long as the hardware is functioning properly and has sufficient power available, telemetry access would always be assured.

3. A capability to load data logging program modules remotely from the Data Center and/or parameter lists that govern various aspects of the data logging operation, e.g., sampling interval times.

These features were incorporated into the system with a combination of software and hardware an important part of which is the forced reset described on page 8.

Failsafe operation is achieved in the following way. The telephone ringing circuit due to an incoming call forces the TIM to reset and restart at a preset memory location.. This response is designed into the hardware. Prior to the reset, detection circuitry informs the CPU of the impending reset. Sufficient time is allowed for the program to set up the necessary conditions that will allow proper continuation of the program at the end of the call. After the reset the CPU executes the ROM resident coding to service the incoming call. One of the reasons for incorporating a forced timeout-restart capability was to prevent the TIM system getting trapped in this coding by a telephone line disconnection. The program immediately interrogates a status register to determine the reason for the restart. If the restart is due to an auto-restart condition, usually because the data logging software has failed in some way, e.g., power was lost for some period of time, the system goes into a standby mode. If the restart was caused by manual intervention, provision was made to use the panel switches to select an address branch. This feature enables an operator to restart the TIM subsystem or to select a specialized subprogram such as a test routine during the initial installation.

Generally the restart will be caused by an incoming telephone call from the Data Center. Once having established that the

telephone ring was the cause, the program allows the modem to answer the call and then waits for an incoming message packet. Upon receipt of the packet, several checks are made such as the parity of each byte and checksum. If the packet is in order, the op-code byte is examined. The contents of the op-code determine the disposition of the contents of the packet as shown in Figure 4b. After the packet has been properly attended to, the program either ackowledges receipt of the packet or replies with the requested data. The program then loops back to await another packet. The last packet type received will normally be a command to terminate the telemetry linkup and return to the data logging function. Appendix C1 outlines the usage of the various types of packets. It is therefore clear that as long as all the telemetry program module resides in the unalterable ROM failsafe, telemetery is assured. On the other hand since the data logging module resides in RAM, it can be readily changed or reloaded.

The procedure for invoking data communication with a TIM is essentially as follows:

Initially a TIM is placed at a site. It is connected to a 12V power source, a telephone jack and the signal lines from the instruments. The TIM is activated (see Appendix E3) then with the use of a portable terminal, the TIM telemetry program (named "X-TIM.TALK") in the Data Center computer is invoked. "X-TIM.TALK" then dials up the TIM and upon achieving a hookup proceeds to transmit message packets with instructions to the TIM to use the packet contents as program parts to be loaded into the RAM. After the data logging software has been completely transmitted and loaded, communication is terminated and the TIM immediately transfers to the data logging mode.

Thereafter routinely, the Data Center system calls up each TIM in turn (usually at 3 a.m. each day) and interrogates them. The data for the preceding data logging period is received from the TIM and stored in a file identified for that station and the origin time for that data set.

All communication between the Data Center computer and a TIM utilizes message packets (Appendix C). Each byte of message package is checked for parity and there is checksum checking of each packet and format adherence at both ends. There is acknowledgement of reception of an error free message packet or a rejection because of an error. If an error occurs, retransmission of the packet is requested and if after several attempts an error free reception is not achieved, the communication is aborted and the TIM is redialed. Several redials will take place before the attempt to attain error free telemetry with any given TIM is aborted.

The data logging module runs independently of the telemetry system and will be described in more detail below as will the telemetry software modules, the Data Center program "X-TIM.TALK" and the TIM telemetry module.

7.2 Data Center - "X-TIM.TALK"

This program runs in the Caltech Seismological Laboratory "PRIME 750" computer. Its function is to dial up each TIM site in turn, and once having a telemetry hookup, to make requests of the TIM. These requests are governed by a control word. Appendix C lists the current requests that may be made.

"X-TIM.TALK" is directed by a station list file (see Figure 2). This list contains a one-line entry for each TIM site. Each entry consists of a 2 character ID, the 5 octal digit control word referred to above, and the telephone number. A simplified flow diagram is shown in Fig. 3. Full details are included in program listing in Appendix J.

7.3 TIM - "X-TIM.COMM*PA

"X-TIM.COMM*PA" is the TIM telemetry software module. It resides in Read-Only-Memory (ROM) and handles all aspects of the communications between the TIM and the Data Center. At any time the TIM is forced to restart as previously discussed, the program module residing in the ROM takes control. This is usually telemetry software.

The telemetry module answers the incoming call, accepts message packets from the data center and responds to the contents of the message packet whether it be a request for data or instructions to be carried out using the contents of the packet. It also controls the modem allowing full two-way communication albeit only one direction at a time.

Figures 4a,b show a simplified flow diagram for "X-TIM.COMM*PA". Complete details are included in the listing in Appendix K.

7.4 TIM - "LOGR"*PA"

Figures 5a-e show a simplified flow diagram of "LOGR". A listing is given in Appendix L.

"LOGR" is a general purpose data logging software module that

resides in RAM. It is loaded via the telemetry function. Each TIM site has its own loadable "LOGR" module residing in the Data Center files so special requirements are easily managed. However, at this time all sites utilize the same general purpose "LOGR".

"LOGR" will accept data on any combination of 8 analog input channels or 4 digital input channels. Sampling intervals of the analog channels are normally governed by the software. Digital data is normally externally clocked and "LOGR" merely accepts the digital data at any time it enters the system. One analog channel is usually used to measure power supply voltage to allow daily surveillance of voltage for those sites utilizing batteries.

The period between samples, i.e., the sampling interval of analog channels is determined by a table, each channel having a separate entry in the table. In this way each channel will be sampled at its own predetermined rate in units of one minute. This table by itself can be reloaded at any time and so sampling intervals can be rapidly and conveniently changed by the Data Center to reflect any change in need. The "LOGR" module itself is not normally changed but if the Data Center becomes aware that the data logging is not proceeding properly upon acquiring the system status of the TIM, it will automatically reload a fresh "LOGR" module. It will also get a program dump prior to the reload so that a postmortem may be made of the failure.

"LOGR" utilizes the data RAM to store the data it acquires in between data dumps to the Data Center. Data is stored in blocks consisting of 6 header words and 58 data words. The 6 header words contain sufficient information so that the integrity of each data block can be relied on even though the TIM may fail or telephone circuits cause unreliable data transmissions.

"LOGR" also allows an installer to monitor any of the data channels on a real time basis as well as the elapsed time in hours and minutes since the previous data dump and clock reset or program load. This can be a valuable tool for establishing the integrity of the data source during the installation of the TIM and instrumentation.

8 DISCUSSION

The CROSS concept is proving to be a reliable method of collecting ultralong period data. For close or nearby sites it offers great convenience since the data automatically appears on the Data Center computer and is usually no more than 24 hours old. For remote sites CROSS has the additional advantages of economical and timely data retrieval.

The TIM hardware has proved to be quite reliable. Units have been operating over a year with no problems and for over six months on an original program load. Reliablity is also enhanced by the absence of any moving parts, particularly a tape recorder with all its potential problems.

In those few circumstances in which the TIM ceased data logging, logging was restored following the next telemetry access and reloading of "LOGR". Occasionally the telephone lines have become noisy making error free telemetry impossible. In these cases, after several attempts at redialing, the system aborts on those stations. When this happens, hopefully adequate service is restored before the data RAM is full but on occasions overflow has occurred first and data has been lost.

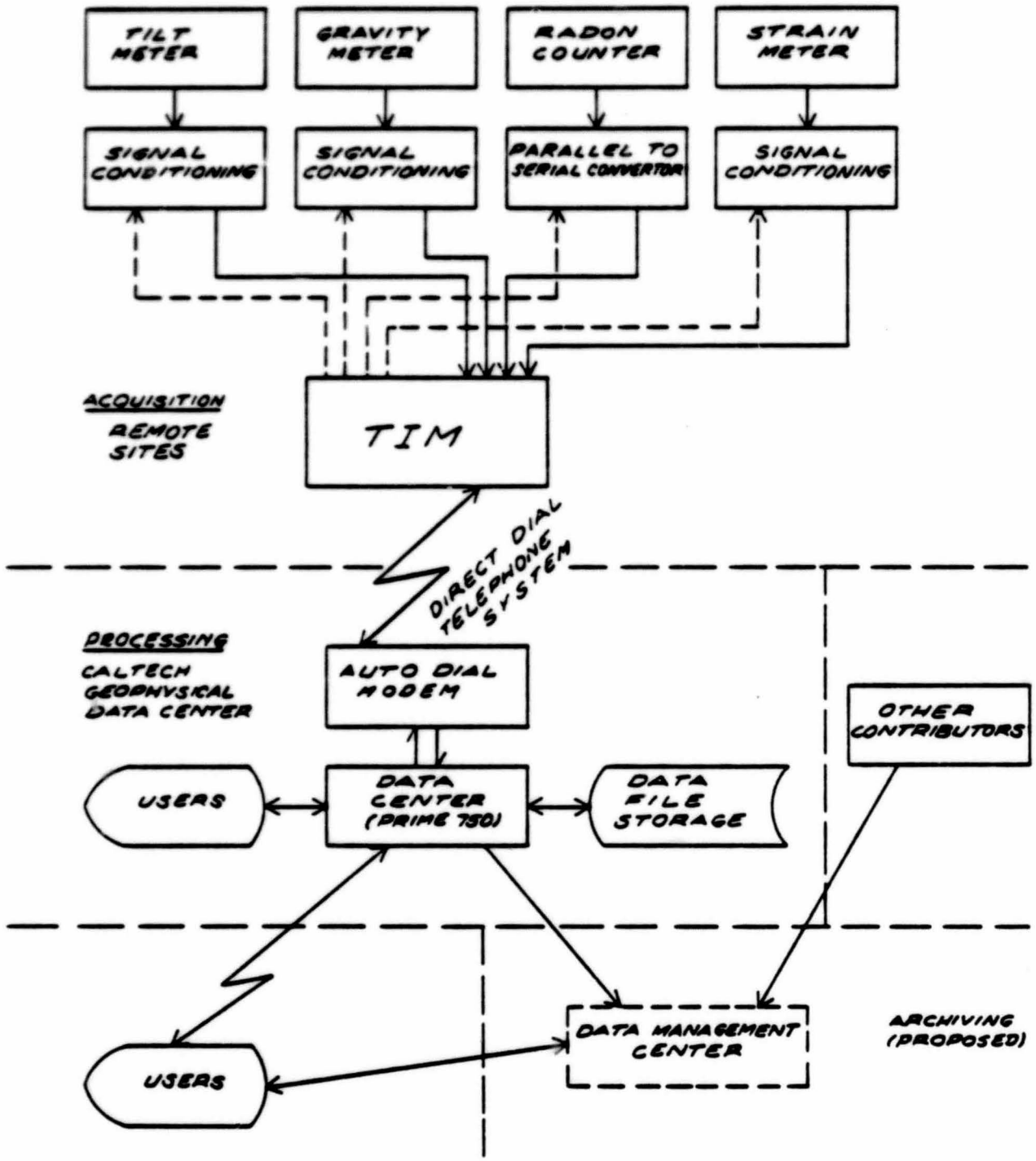


Figure 1. Block diagram of CROSS

KR260037956415
RO260035772487
PD2600318052731614
HO2600314086376142
BR2600318052701250
SL260034499398
* AZ4000317147634067
* VL2600318059442944
* BC260038992444
* VY2600318059443016
* FR2600318057248271

Figure 2. Station Control List
(*indicates not active)

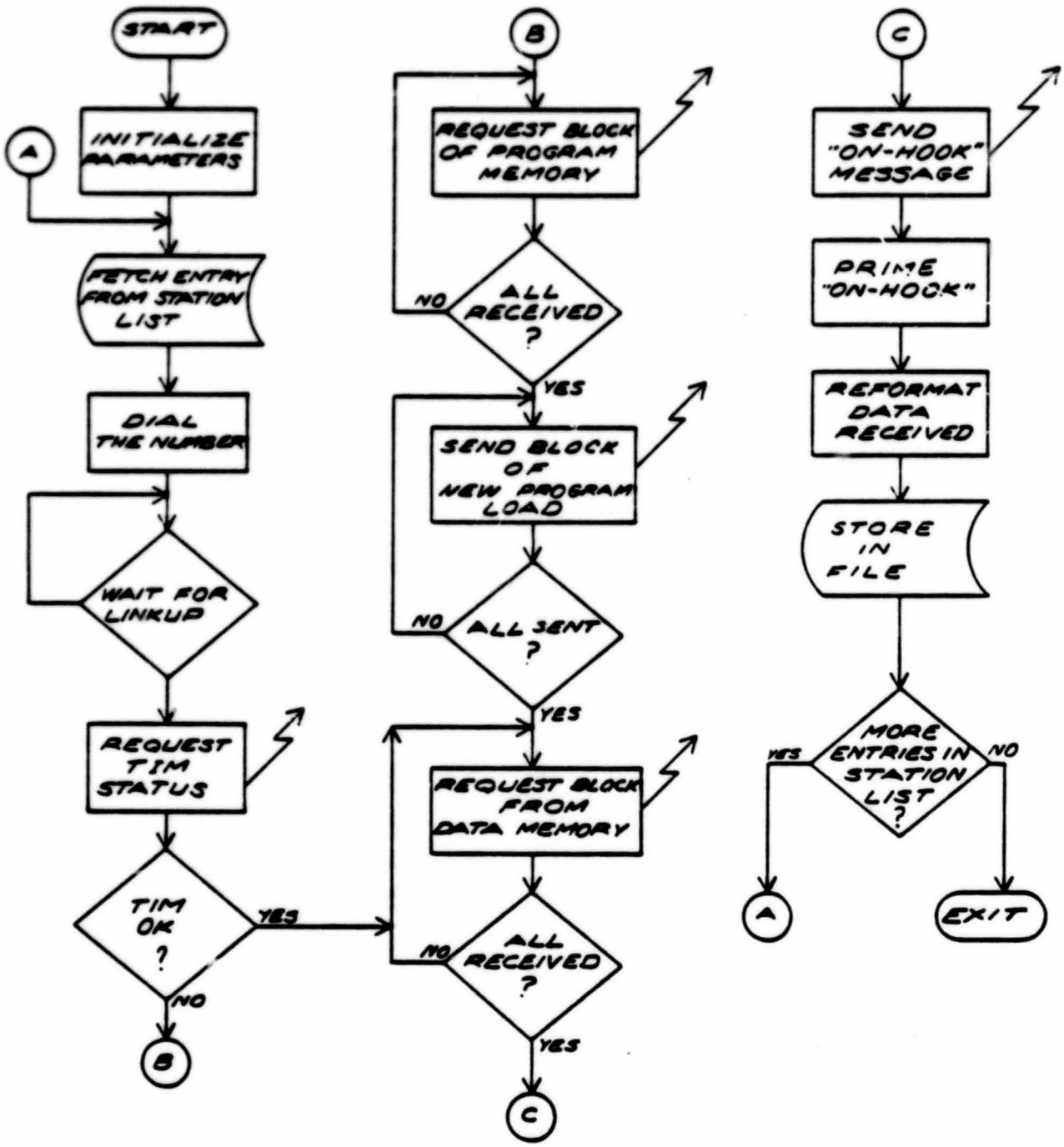


Figure 3. Data Center - TIM interrogation sequence
(General flow chart of "X-TIM.TALK")

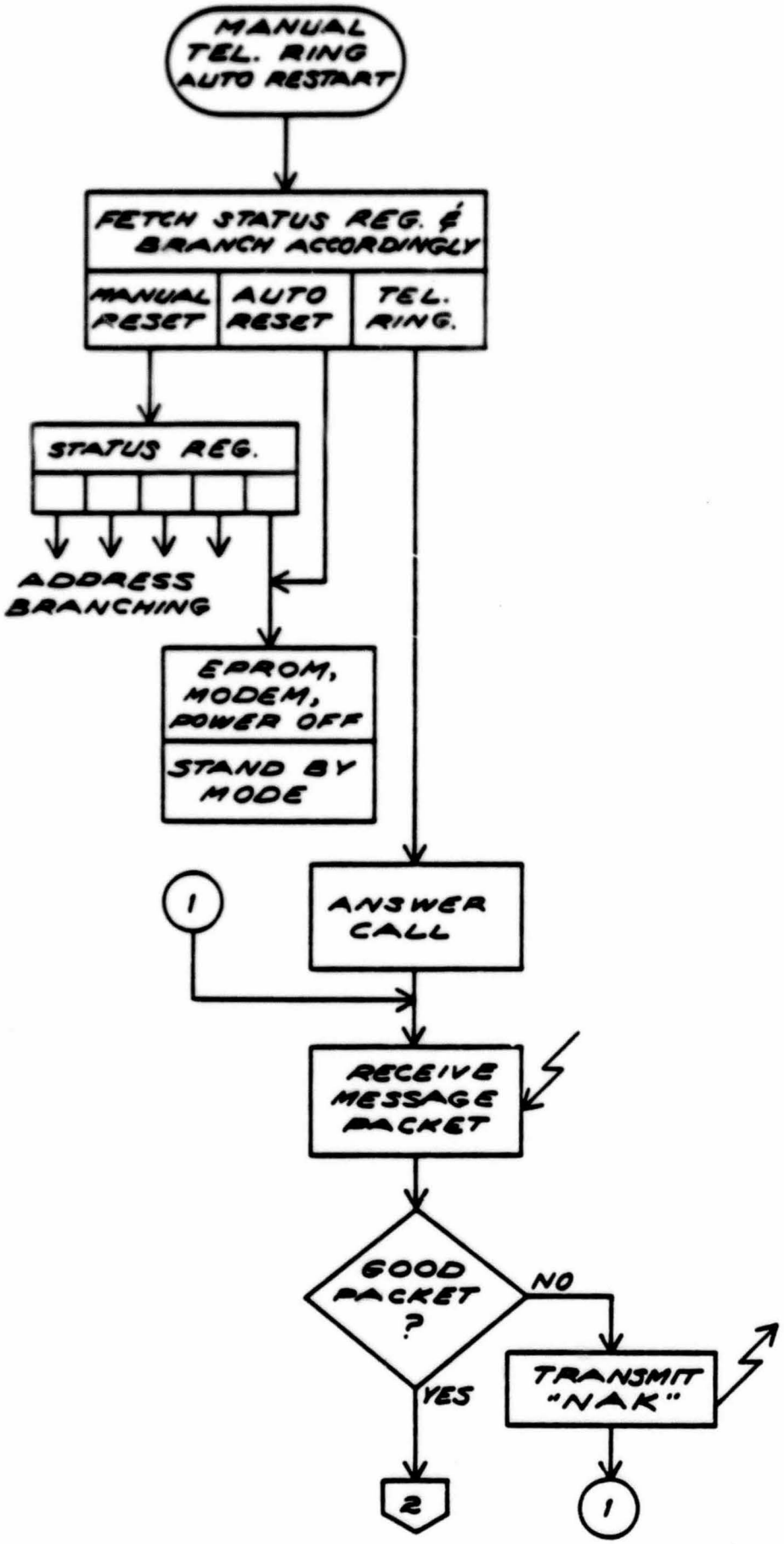


Figure 4a. Flow chart of TI:1 telemetry software module.

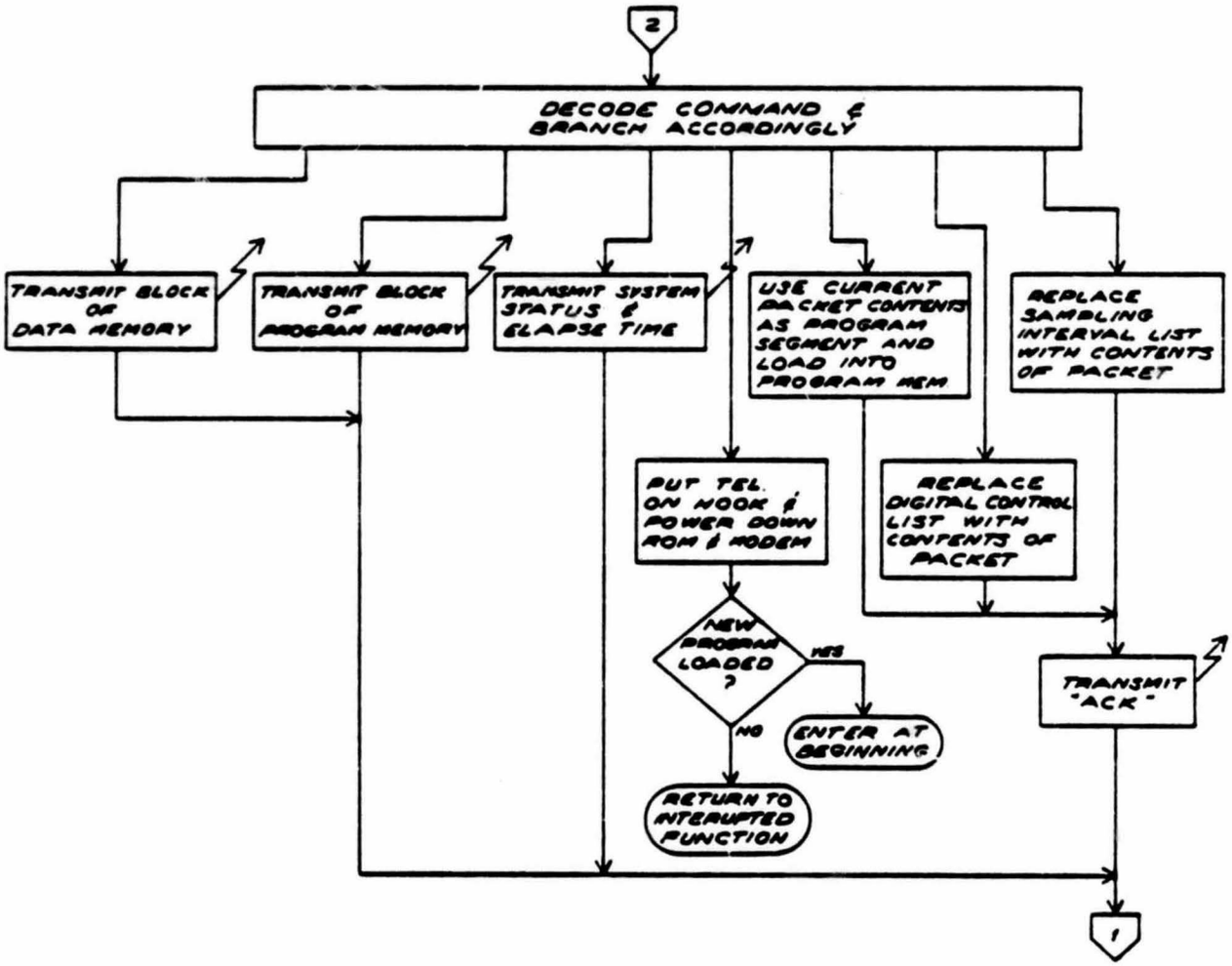

2
DECODE COMMAND & BRANCH ACCORDINGLY
TRANSMIT BLOCK OF DATA MEMORY
TRANSMIT BLOCK OF PROGRAM MEMORY
TRANSMIT SYSTEM STATUS & ELAPSE TIME
USE CURRENT PACKET CONTENTS AS PROGRAM SEGMENT AND LOAD INTO PROGRAM MEM
REPLACE SAMPLING INTERVAL LIST WITH CONTENTS OF PACKET
PUT TEL. ON HOOK & POWER DOWN ROM & MODEM
REPLACE DIGITAL CONTROL LIST WITH CONTENTS OF PACKET
NEW PROGRAM LOADED ?
YES
NO
ENTER AT BEGINNING
RETURN TO INTERUPTED FUNCTION
TRANSMIT "ACK"
1

Figure 4b.

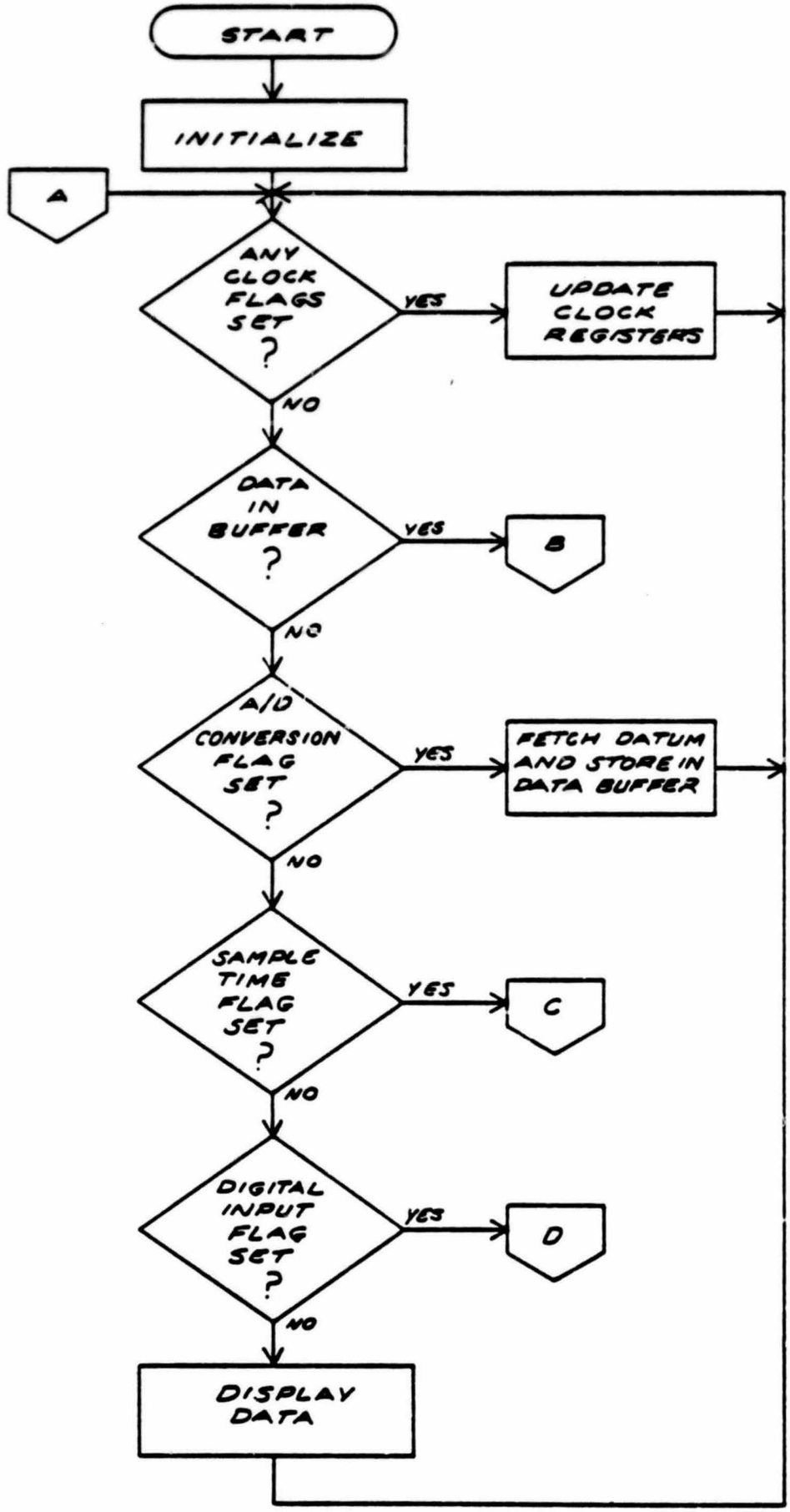


Figure 5a. Basic flow chart of TIM data logging software module.

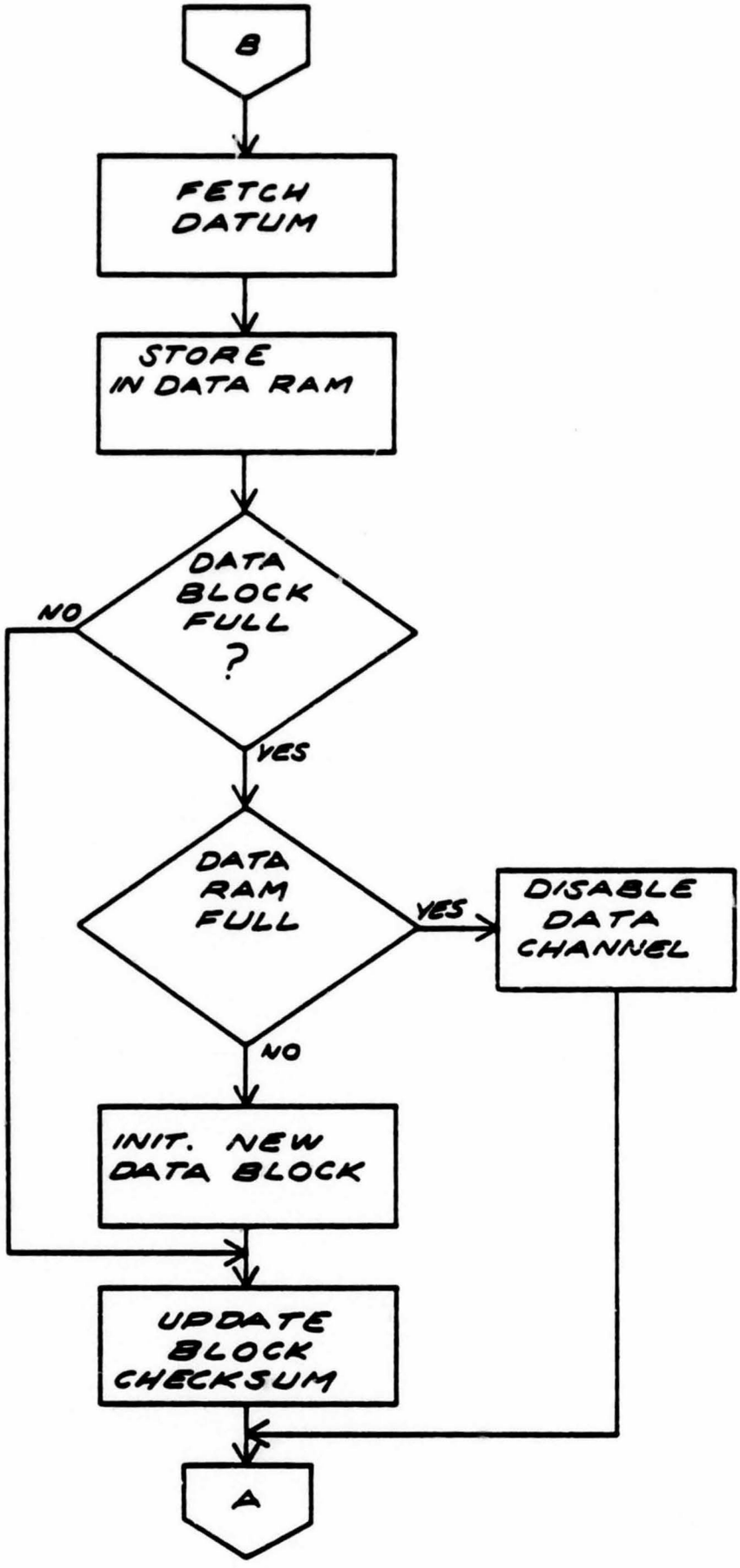

B
FETCH DATUM
STORE IN DATA RAM
DATA BLOCK FULL ?
NO
YES
DATA RAM FULL
YES
DISABLE DATA CHANNEL
NO
INIT. NEW DATA BLOCK
UPDATE BLOCK CHECKSUM
A

Figure 5b.

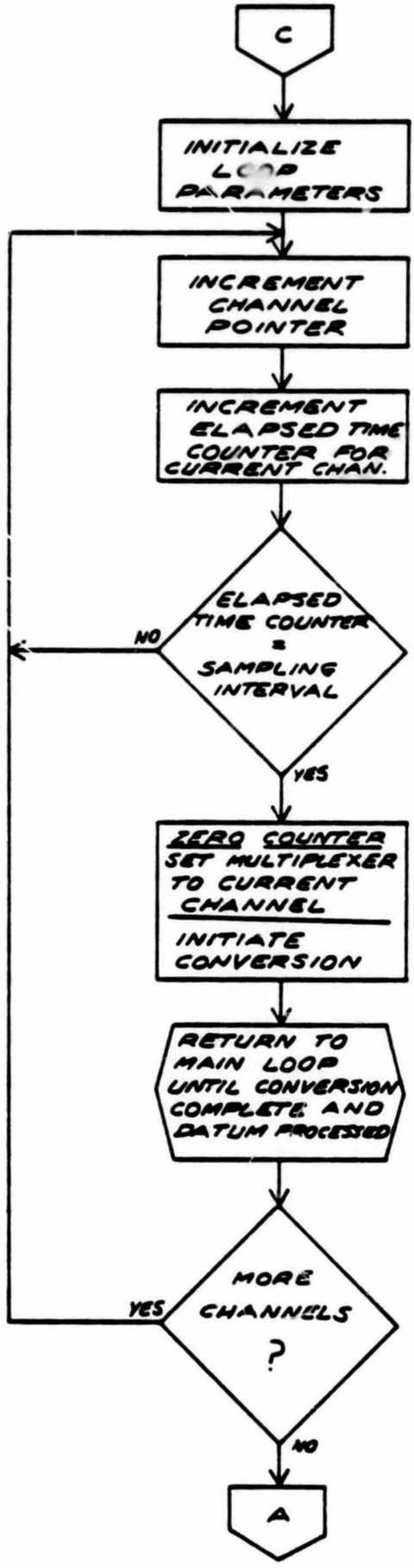

C
INITIALIZE LOOP PARAMETERS
INCREMENT CHANNEL POINTER
INCREMENT ELAPSED TIME COUNTER FOR CURRENT CHAN.
ELAPSED TIME COUNTER = SAMPLING INTERVAL
NO
YES
ZERO COUNTER
SET MULTIPLEXER TO CURRENT CHANNEL
INITIATE CONVERSION
RETURN TO MAIN LOOP UNTIL CONVERSION COMPLETE AND DATUM PROCESSED
MORE CHANNELS ?
YES
NO
A

Figure 5c.

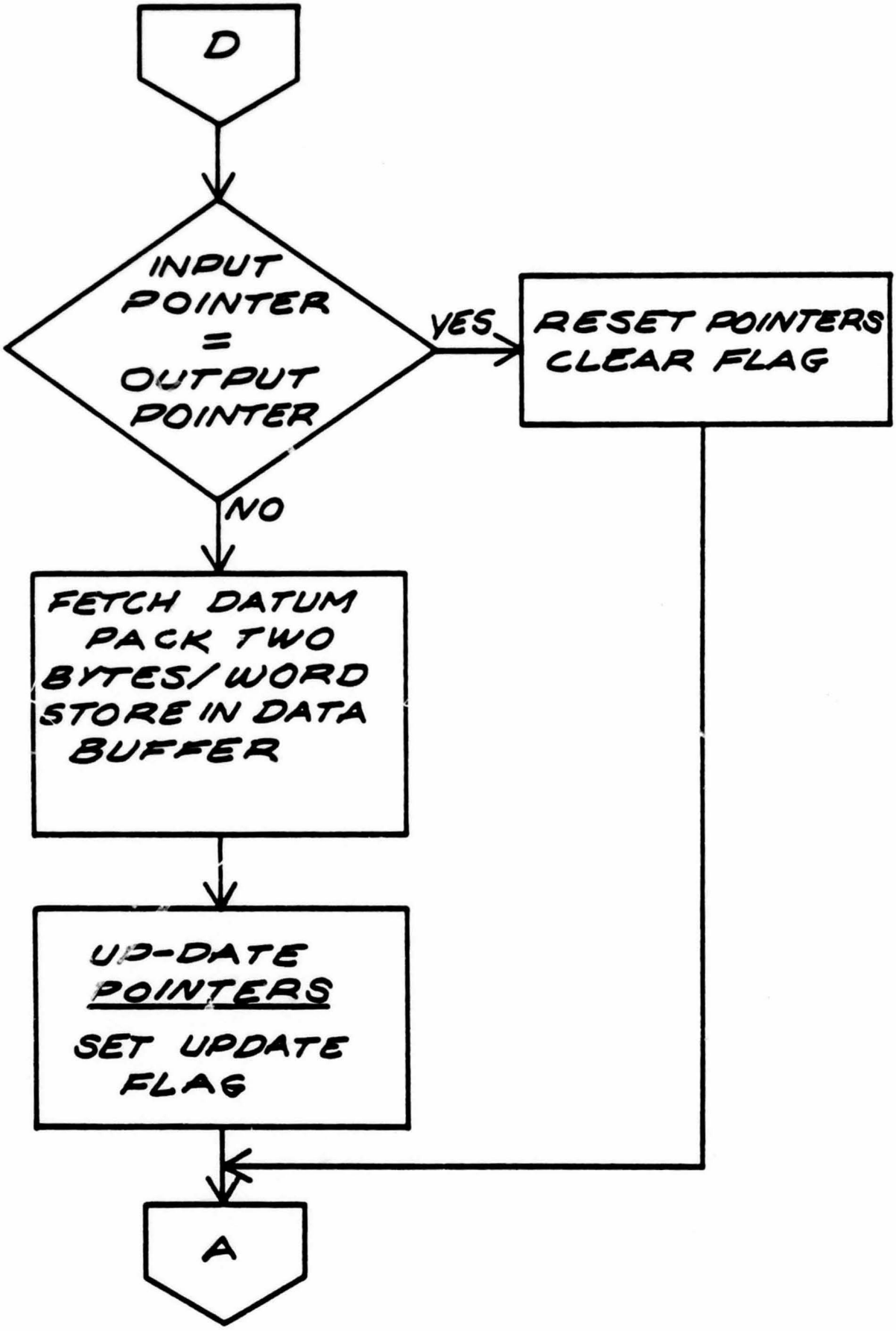

D
INPUT POINTER = OUTPUT POINTER
YES
RESET POINTERS CLEAR FLAG
NO
FETCH DATUM PACK TWO BYTES/WORD STORE IN DATA BUFFER
UP-DATE POINTERS
SET UPDATE FLAG
A

Figure 5d.

Appendix A

GLOSSARY OF ACRONYMS AND TERMS

ASCII	American Standard Code for Information Interchange
BYTE	6 to 8 bit data word
CPU	Central Processing Unit, commonly used terms that refers to the computing section of a computer.
CROSS	Caltech Remote Observatory Support System
DAA	Data Access Arrangement, required by telephone company regulations for attaching non-FCC licensed equipment to telephone lines.
EPROM	Erasable Programmable Read Only Memory.
RAM	Random Access Memory, for storage of computer program and/or data.
ROM	Read Only Memory
TIM	Telemetry Interface Module
UART	Universal Asynchronous Receiver-Transmitter, a key element in any two wire digital communications transmission system.

Appendix B

List of TIM Stations as of October 31, 1979

Id	SITE LOCATION	# CHAN. IN USE	TYPE of MEASUREMENTS	PRINCIPAL INVESTIGATOR
KR	Kresge Lab.	6	tilt,gravimeter	Test site
RO	Caltech campus	6	tilt	T. Ahrens
PD	Palmdale	8	tellurics	T.A. Madden
HO	Hollister	8	tellurics	T.A. Madden
BC	Buck Canyon	1	strain	B. Clark
VL	Valyermo	1	strain	B. Clark
BR	Bouquet Reservoir	1	Gravimeter	L. Teng
VY	Valyermo	1	water well	D. Lamar/K. McNally
AZ	Anza	1	water well	D. Lamar/K. McNally

Appendix C

PRIME - TIM MESSAGE PACKET TYPES

Name	Description
STATUS	Send TIM status to Data Center. The reply packet also includes accumulated time since last TIM clock reset.
LOAD-PROG	Packet contains segment of a program to be loaded into the TIM program memory.
LOAD-SRL	Packet contains list for the TIM sampling interval table.
LOAD-DCL	Packet contains list for the TIM Digital Control table.
DUMP-PROG	TIM is to send a block of its program memory as defined by the data words.
DUMP-DATA	TIM is to send a block of the data memory as defined by the data words.
CEASE	The TIM unit is to put the telephone line on-hook and return to data logging.

Appendix C

PRIME - TIM MESSAGE PACKET FORMAT

Byte Use	Comment
Sync Code	Always first in each packet
# bytes in packet	Size of packet
Station Id	2 6-bit ASCII characters
OP-Code	Identifies nature of packet
Data	
:	
:	
Checksum	
EOM	End of message packet

Appendix C

SPECIFIC PACKET FORMATS
(ALL SINGLE BYTE WORDS = 100+BYTE, E.G. EOM=103)

Data Center to X-TIM

REQUEST FOR SYSTEM STATUS
- # BYTES IN PACKET (=OR$128)
- STATION ID #
- OP CODE (142)
- CHECKSUM
- EOM

REQUEST FOR PROGRAM OR DATA DUMP
- SYNC CODE
- # BYTES IN PACKET
- STATION ID
- OP CODE (140 OR 141)
- FWA OF BLOCK TO BE SENT
- # WORDS TO BE SENT
- CHECKSUM
- EOM

TRANSMIT PROGRAM OR LIST
- SYNC CODE
- # BYTES IN PACKET
- STATION ID
- OP CODE (143-146)
- DATA
  :
  :
- CHECKSUM
- EOM

TERMINATE
- SYNC CODE
- # BYTES IN PACKET
- STATION ID
- OP CODE (147)
- # SECS. IN CURR. MIN.
- ACTION WORD
- CHECKSUM
- EOM

TIM to Data Center
(12-bit word transmission in HI, LO bytes)

System Status to Data Center

SYNC CODE
STATION ID #
TIME - HI ORDER
TIME - LO ORDER
SYSTEM STATUS WORD
DATA BLOCK POINTER
CHECKSUM
EOM

Data Packets to Data Center

SYNC CODE
STATION ID #
FWA OF BLOCK
WORDS IN BLOCK
DATA
CHECKSUM
EOM

Appendix D

LIST OF TIM DATA FILES UP TO NOVEMBER 6, 1979

File Name Format

ABXX.YYY.ZZ

AB Two character Id. code
XX Year of origin time of data
YYY Day of origin time of data
ZZ GMT of origin time of data

Appendix D

BC79.109.22
BC79.110.07
BC79.111.07
BC79.112.07
BC79.113.07
BC79.114.07
BC79.115.07
BC79.116.07
BC79.117.07
BC79.118.07
BC79.119.07
BC79.120.07
BC79.121.07
BC79.123.06
BC79.124.06
BC79.128.06
BC79.129.06
BC79.130.06
BC79.131.06
BC79.132.06
BC79.133.06
BC79.134.06
BC79.135.06
BC79.136.06
BC79.137.06
BC79.138.06
BC79.139.06
BC79.140.06
BC79.141.06
BC79.142.06
BC79.143.06
BC79.144.06
BC79.145.06
BC79.146.06
BC79.147.06
BC79.148.06
BC79.149.06
BC79.150.07
BC79.151.06
BC79.152.08
BC79.153.05
BC79.154.06
BC79.155.06
BC79.156.06
BC79.157.06
BC79.158.06
BC79.159.06
BC79.171.06
BC79.172.06
BC79.173.18
BC79.175.06
BC79.176.06
BC79.177.06
BC79.178.06
BC79.179.06
BC79.180.06
BC79.181.19
BC79.182.06
BC79.183.06
BC79.184.06
BC79.185.18
BC79.187.06
BC79.188.06
BC79.189.10
BC79.190.10
BC79.191.10
BC79.192.10
BC79.193.10
BC79.194.10
BC79.195.23
BC79.196.10
BC79.197.10
BC79.198.10
BC79.199.10
BC79.200.10
BC79.201.10
BC79.202.10
BC79.203.10
BC79.204.10
BC79.205.10
BC79.206.10
BC79.207.10
BC79.208.10
BC79.210.18
BR79.228.23
BR79.229.10
BR79.230.10
BR79.231.10
BR79.232.10
BR79.233.10
BR79.234.10
BR79.235.10
BR79.236.10
BR79.237.10
BR79.238.10
BR79.239.10
BR79.240.10
BR79.241.10
BR79.243.10
BR79.244.10
BR79.245.10
BR79.246.10
BR79.247.10
BR79.248.10
BR79.249.10
BR79.250.18
BR79.252.10
BR79.253.10
BR79.254.10
BR79.255.10
BR79.256.10
BR79.257.10
BR79.258.10
BR79.259.10
BR79.260.10
BR79.261.10
BR79.262.10
BR79.263.10
BR79.264.10
BR79.265.10
BR79.266.10
BR79.267.10
BR79.269.10
BR79.270.10
BR79.271.18
BR79.273.10
BR79.274.10
BR79.275.21
BR79.276.10
BR79.277.10
BR79.278.10
BR79.279.10
BR79.280.10
BR79.281.10
BR79.282.10
BR79.283.10
BR79.284.10
BR79.285.10
BR79.287.03
BR79.288.10
BR79.289.10
BR79.290.10
BR79.291.11
BR79.293.16
BR79.295.20
BR79.297.02
BR79.298.10
BR79.299.10
BR79.302.18
BR79.304.06
BR79.305.11
BR79.306.11
BR79.307.11
BR79.308.11
FR79.248.02
FR79.249.20
FR79.254.00
GS79.023.00
GS79.023.06
GS79.024.20
GS79.025.08
GS79.026.07
GS79.027.19
GS79.028.07
GS79.031.19
GS79.031.22
HO79.052.23
HO79.053.07
HO79.054.07
HO79.055.07
HO79.056.07
HO79.057.07
HO79.058.07
HO79.059.07
HO79.060.18
HO79.061.07
HO79.062.07
HO79.063.07
HO79.064.07
HO79.065.07
HO79.066.07
HO79.067.07
HO79.068.07
HO79.069.07
HO79.070.07
HO79.071.07
HO79.072.21
HO79.073.07
HO79.074.07
HO79.075.07
HO79.076.07
HO79.077.07
HO79.078.07
HO79.079.20
HO79.080.07
HO79.081.07
HO79.082.07
HO79.083.07
HO79.084.07
HO79.085.07

Appendix D

H079.086.07	H079.182.06	H079.240.10	H079.290.10
H079.087.07	H079.183.06	H079.241.10	H079.291.10
H079.088.07	H079.184.06	H079.242.10	H079.292.10
H079.089.07	H079.185.18	H079.243.10	H079.293.16
H079.090.07	H079.187.06	H079.244.10	H079.295.19
H079.091.07	H079.188.06	H079.245.10	H079.297.02
H079.092.07	H079.189.10	H079.246.10	H079.298.10
H079.093.07	H079.190.10	H079.247.10	H079.299.10
H079.094.07	H079.191.10	H079.248.10	H079.302.18
H079.095.07	H079.192.10	H079.249.10	H079.304.06
H079.096.07	H079.193.10	H079.250.10	H079.305.11
H079.097.20	H079.194.10	H079.251.10	H079.306.11
H079.099.07	H079.195.23	H079.252.10	H079.307.11
H079.100.07	H079.196.10	H079.253.10	H079.308.11
H079.101.07	H079.197.10	H079.254.10	KR78.353.03
H079.102.07	H079.204.21	H079.255.10	KR78.354.02
H079.103.07	H079.205.10	H079.256.10	KR78.355.02
H079.104.07	H079.206.10	H079.257.10	KR78.355.08
H079.105.07	H079.208.22	H079.258.10	KR78.356.02
H079.106.07	H079.209.18	H079.259.10	KR78.357.02
H079.107.07	H079.210.18	H079.260.10	KR78.358.02
H079.108.07	H079.212.00	H079.261.10	KR78.359.02
H079.109.07	H079.212.20	H079.261.20	KR78.360.02
H079.110.07	H079.213.10	H079.262.10	KR78.361.02
H079.111.07	H079.214.10	H079.263.10	KR78.362.02
H079.112.07	H079.215.20	H079.264.10	KR78.363.02
H079.113.07	H079.216.23	H079.265.10	KR78.364.02
H079.114.07	H079.217.10	H079.266.10	KR78.365.02
H079.115.07	H079.218.10	H079.267.10	KR78.366.02
H079.116.07	H079.219.10	H079.268.10	KR79.002.02
H079.117.07	H079.220.10	H079.269.10	KR79.003.00
H079.118.07	H079.221.10	H079.270.10	KR79.003.02
H079.119.07	H079.222.10	H079.271.18	KR79.004.02
H079.165.21	H079.223.10	H079.272.10	KR79.005.02
H079.166.06	H079.224.10	H079.273.10	KR79.006.02
H079.167.06	H079.225.10	H079.274.10	KR79.007.02
H079.168.06	H079.226.17	H079.275.21	KR79.008.02
H079.169.06	H079.227.10	H079.276.10	KR79.009.20
H079.170.06	H079.228.10	H079.277.10	KR79.010.10
H079.170.17	H079.229.10	H079.278.10	KR79.011.04
H079.171.06	H079.230.10	H079.279.10	KR79.012.07
H079.172.06	H079.231.10	H079.280.10	KR79.013.07
H079.173.18	H079.232.10	H079.281.10	KR79.014.07
H079.175.06	H079.233.10	H079.282.10	KR79.015.07
H079.176.06	H079.234.10	H079.283.10	KR79.016.07
H079.177.06	H079.235.10	H079.284.10	KR79.017.07
H079.178.06	H079.236.10	H079.285.10	KR79.018.07
H079.179.06	H079.237.10	H079.287.02	KR79.019.07
H079.180.06	H079.238.10	H079.288.10	KR79.020.07
H079.181.19	H079.239.10	H079.289.10	KR79.021.07

Appendix D

KR79.022.07	KR79.072.21	KR79.128.06	KR79.177.06
KR79.023.06	KR79.073.07	KR79.129.06	KR79.178.06
KR79.024.07	KR79.074.07	KR79.130.06	KR79.179.06
KR79.025.08	KR79.075.07	KR79.131.06	KR79.180.06
KR79.026.07	KR79.076.07	KR79.132.06	KR79.181.19
KR79.027.07	KR79.077.07	KR79.133.06	KR79.182.06
KR79.028.07	KR79.078.07	KR79.134.06	KR79.183.06
KR79.029.07	KR79.080.07	KR79.135.06	KR79.184.06
KR79.030.07	KR79.081.07	KR79.136.06	KR79.185.18
KR79.030.07	KR79.082.07	KR79.137.06	KR79.187.06
KR79.032.07	KR79.083.07	KR79.138.06	KR79.188.06
KR79.033.07	KR79.084.07	KR79.139.06	KR79.189.10
KR79.034.07	KR79.085.07	KR79.140.06	KR79.190.10
KR79.035.07	KR79.086.07	KR79.141.06	KR79.191.10
KR79.036.07	KR79.087.07	KR79.142.06	KR79.192.10
KR79.037.07	KR79.088.07	KR79.143.06	KR79.193.10
KR79.038.07	KR79.089.07	KR79.144.06	KR79.194.10
KR79.039.07	KR79.090.07	KR79.145.06	KR79.195.23
KR79.040.07	KR79.091.07	KR79.146.06	KR79.196.10
KR79.042.20	KR79.092.07	KR79.147.06	KR79.197.10
KR79.043.21	KR79.093.07	KR79.148.06	KR79.198.10
KR79.044.01	KR79.094.07	KR79.149.06	KR79.199.10
KR79.044.07	KR79.095.07	KR79.150.07	KR79.200.10
KR79.045.07	KR79.096.07	KR79.151.06	KR79.201.10
KR79.046.07	KR79.097.19	KR79.152.07	KR79.202.10
KR79.047.07	KR79.099.07	KR79.153.04	KR79.203.10
KR79.048.07	KR79.100.07	KR79.154.06	KR79.204.10
KR79.049.07	KR79.101.07	KR79.155.06	KR79.205.10
KR79.050.07	KR79.102.07	KR79.156.06	KR79.206.10
KR79.051.07	KR79.103.07	KR79.157.06	KR79.207.10
KR79.052.07	KR79.104.07	KR79.158.06	KR79.208.10
KR79.053.07	KR79.105.07	KR79.159.06	KR79.210.10
KR79.054.07	KR79.106.07	KR79.160.06	KR79.211.20
KR79.055.07	KR79.107.07	KR79.161.06	KR79.212.19
KR79.056.07	KR79.108.07	KR79.162.06	KR79.213.10
KR79.057.07	KR79.109.07	KR79.163.06	KR79.214.10
KR79.058.07	KR79.110.07	KR79.164.06	KR79.215.10
KR79.059.07	KR79.111.07	KR79.164.19	KR79.217.10
KR79.060.18	KR79.112.07	KR79.165.06	KR79.218.10
KR79.061.07	KR79.113.07	KR79.165.19	KR79.219.10
KR79.062.07	KR79.114.07	KR79.166.06	KR79.220.10
KR79.063.07	KR79.115.07	KR79.167.06	KR79.221.10
KR79.064.07	KR79.116.07	KR79.168.06	KR79.222.10
KR79.065.07	KR79.117.07	KR79.169.06	KR79.223.10
KR79.066.07	KR79.118.07	KR79.170.06	KR79.224.10
KR79.067.07	KR79.119.07	KR79.171.06	KR79.225.10
KR79.068.07	KR79.120.07	KR79.172.06	KR79.227.10
KR79.069.07	KR79.121.07	KR79.173.18	KR79.228.10
KR79.070.07	KR79.123.06	KR79.175.06	KR79.229.10
KR79.071.07	KR79.124.06	KR79.176.06	KR79.230.10

Appendix D

KR79.231.10	KR79.280.10	PD79.012.07	PD79.062.07
KR79.232.10	KR79.281.10	PD79.013.07	PD79.063.07
KR79.233.10	KR79.282.10	PD79.014.07	PD79.064.07
KR79.234.10	KR79.283.10	PD79.015.07	PD79.065.07
KR79.235.10	KR79.284.10	PD79.016.07	PD79.066.07
KR79.236.10	KR79.285.10	PD79.017.07	PD79.067.07
KR79.237.10	KR79.287.02	PD79.018.07	PD79.068.07
KR79.238.10	KR79.288.10	PD79.019.07	PD79.069.07
KR79.239.10	KR79.289.10	PD79.020.07	PD79.070.07
KR79.240.10	KR79.290.10	PD79.021.07	PD79.071.07
KR79.241.10	KR79.291.10	PD79.022.07	PD79.072.21
KR79.242.10	KR79.292.10	PD79.023.06	PD79.073.07
KR79.243.10	KR79.293.16	PD79.024.07	PD79.074.07
KR79.244.10	KR79.295.19	PD79.025.08	PD79.075.07
KR79.245.10	KR79.297.01	PD79.026.07	PD79.076.07
KR79.246.10	KR79.298.10	PD79.027.07	PD79.077.07
KR79.247.10	KR79.299.10	PD79.028.07	PD79.078.07
KR79.248.10	KR79.302.18	PD79.029.07	PD79.079.20
KR79.249.10	KR79.304.06	PD79.030.07	PD79.080.07
KR79.250.10	KR79.305.11	PD79.030.07	PD79.081.07
KR79.251.10	KR79.306.11	PD79.032.07	PD79.082.07
KR79.252.10	KR79.307.11	PD79.033.07	PD79.083.07
KR79.253.10	KR79.308.11	PD79.034.07	PD79.084.07
KR79.254.10	PD78.353.06	PD79.035.07	PD79.085.07
KR79.255.10	PD78.354.02	PD79.036.07	PD79.086.07
KR79.256.03	PD78.355.02	PD79.037.07	PD79.087.07
KR79.256.10	PD78.355.08	PD79.038.07	PD79.088.07
KR79.257.10	PD78.355.22	PD79.039.07	PD79.089.07
KR79.258.10	PD78.356.02	PD79.040.07	PD79.090.07
KR79.259.10	PD78.357.02	PD79.042.20	PD79.091.07
KR79.260.10	PD78.358.02	PD79.043.21	PD79.092.07
KR79.261.10	PD78.359.02	PD79.044.01	PD79.093.07
KR79.262.10	PD78.360.02	PD79.044.07	PD79.094.07
KR79.263.10	PD78.361.02	PD79.045.07	PD79.095.07
KR79.264.10	PD78.362.02	PD79.046.07	PD79.096.07
KR79.265.10	PD78.363.02	PD79.047.07	PD79.097.20
KR79.266.10	PD78.364.02	PD79.048.07	PD79.099.07
KR79.267.10	PD78.365.02	PD79.049.07	PD79.100.07
KR79.268.10	PD78.366.02	PD79.050.07	PD79.101.07
KR79.269.10	PD79.002.02	PD79.051.07	PD79.102.07
KR79.270.10	PD79.003.00	PD79.052.07	PD79.103.07
KR79.271.17	PD79.003.02	PD79.053.07	PD79.104.07
KR79.272.10	PD79.004.02	PD79.054.07	PD79.105.07
KR79.273.10	PD79.005.02	PD79.055.07	PD79.106.07
KR79.274.10	PD79.006.02	PD79.056.07	PD79.107.07
KR79.275.20	PD79.007.02	PD79.057.07	PD79.108.07
KR79.276.10	PD79.008.02	PD79.058.07	PD79.109.07
KR79.277.10	PD79.009.20	PD79.059.07	PD79.110.07
KR79.278.10	PD79.010.10	PD79.060.18	PD79.111.07
KR79.279.10	PD79.011.04	PD79.061.07	PD79.112.07

PD79.113.07
PD79.114.07
PD79.115.07
PD79.116.07
PD79.117.07
PD79.118.07
PD79.119.07
PD79.120.07
PD79.121.07
PD79.123.06
PD79.124.06
PD79.128.06
PD79.129.06
PD79.130.06
PD79.131.06
PD79.132.06
PD79.133.06
PD79.134.06
PD79.135.06
PD79.136.06
PD79.137.06
PD79.138.06
PD79.139.06
PD79.140.06
PD79.141.06
PD79.142.06
PD79.143.06
PD79.144.06
PD79.145.06
PD79.146.06
PD79.147.06
PD79.148.06
PD79.149.06
PD79.150.07
PD79.151.06
PD79.152.07
PD79.153.05
PD79.154.06
PD79.155.06
PD79.156.06
PD79.157.06
PD79.158.06
PD79.159.06
PD79.160.06
PD79.161.06
PD79.162.06
PD79.163.06
PD79.164.06
PD79.165.06
PD79.165.19
PD79.166.06
PD79.167.06
PD79.168.06
PD79.169.06
PD79.170.06
PD79.171.06
PD79.172.06
PD79.173.17
PD79.175.06
PD79.176.06
PD79.177.06
PD79.178.06
PD79.179.06
PD79.180.06
PD79.181.19
PD79.182.06
PD79.183.06
PD79.184.06
PD79.185.18
PD79.187.06
PD79.188.06
PD79.189.10
PD79.190.10
PD79.191.10
PD79.192.10
PD79.193.10
PD79.194.10
PD79.195.23
PD79.196.10
PD79.197.10
PD79.198.10
PD79.199.10
PD79.200.10
PD79.201.10
PD79.202.10
PD79.203.10
PD79.204.10
PD79.205.10
PD79.206.10
PD79.207.10
PD79.208.10
PD79.209.18
PD79.210.16
PD79.211.20
PD79.212.20
PD79.213.10
PD79.214.10
PD79.215.10
PD79.216.22
PD79.217.10
PD79.218.10
PD79.219.10
PD79.220.10
PD79.221.10
PD79.222.10
PD79.223.10
PD79.224.10
PD79.225.10
PD79.226.17
PD79.227.10
PD79.228.10
PD79.229.10
PD79.230.10
PD79.231.10
PD79.232.10
PD79.233.10
PD79.234.10
PD79.235.10
PD79.236.10
PD79.237.10
PD79.238.10
PD79.239.10
PD79.240.10
PD79.241.10
PD79.242.10
PD79.243.10
PD79.244.10
PD79.245.10
PD79.246.10
PD79.247.10
PD79.248.10
PD79.249.10
PD79.250.10
PD79.251.10
PD79.252.10
PD79.253.10
PD79.254.10
PD79.255.10
PD79.256.10
PD79.257.10
PD79.258.10
PD79.259.10
PD79.260.10
PD79.261.10
PD79.262.10
PD79.263.10
PD79.264.10
PD79.265.10
PD79.266.10
PD79.267.10
PD79.268.10
PD79.270.10
PD79.271.18
PD79.272.10
PD79.273.10
PD79.274.10
PD79.275.20
PD79.276.10
PD79.277.10
PD79.278.10
PD79.279.10
PD79.280.10
PD79.281.10
PD79.282.10
PD79.283.10
PD79.284.10
PD79.285.10
PD79.287.02
PD79.288.10
PD79.289.10
PD79.290.10
PD79.291.10
PD79.292.10
PD79.293.16
PD79.295.19
PD79.297.02
PD79.298.10
PD79.299.10
PD79.302.18
PD79.304.06
PD79.305.11
PD79.306.11
PD79.307.11
PD79.308.11
RO79.040.23
RO79.042.20
RO79.043.21
RO79.044.01
RO79.044.07
RO79.045.07
RO79.046.07
RO79.047.07
RO79.048.07
RO79.049.07
RO79.050.07
RO79.051.07
RO79.052.07
RO79.053.07
RO79.054.07
RO79.055.07

Appendix D

R079.056.07	R079.107.07	R079.161.06	R079.212.19
R079.057.07	R079.108.07	R079.162.06	R079.213.10
R079.058.07	R079.109.07	R079.163.06	R079.214.10
R079.059.07	R079.110.07	R079.164.01	R079.215.10
R079.060.18	R079.111.07	R079.164.06	R079.217.10
R079.061.07	R079.112.07	R079.165.06	R079.218.10
R079.062.07	R079.113.07	R079.165.19	R079.219.10
R079.063.07	R079.114.07	R079.166.06	R079.220.10
R079.064.07	R079.115.07	R079.167.06	R079.221.10
R079.065.07	R079.116.07	R079.168.06	R079.222.10
R079.066.07	R079.117.07	R079.169.06	R079.223.10
R079.067.07	R079.118.07	R079.170.06	R079.224.10
R079.068.07	R079.119.07	R079.171.06	R079.225.10
R079.069.07	R079.120.07	R079.172.06	R079.227.10
R079.070.07	R079.121.07	R079.173.17	R079.228.10
R079.071.07	R079.123.06	R079.175.06	R079.229.10
R079.072.21	R079.124.06	R079.176.06	R079.230.10
R079.073.07	R079.128.06	R079.177.06	R079.231.10
R079.074.07	R079.129.06	R079.178.06	R079.232.10
R079.075.07	R079.130.06	R079.179.06	R079.233.10
R079.076.07	R079.131.06	R079.180.06	R079.234.10
R079.077.07	R079.132.06	R079.181.19	R079.235.10
R079.078.07	R079.133.06	R079.182.06	R079.236.10
R079.080.07	R079.134.06	R079.183.06	R079.237.10
R079.081.07	R079.135.06	R079.184.06	R079.238.10
R079.082.07	R079.136.06	R079.185.18	R079.239.10
R079.083.07	R079.137.06	R079.187.06	R079.240.10
R079.084.07	R079.138.06	R079.188.06	R079.241.10
R079.085.07	R079.139.06	R079.189.10	R079.242.10
R079.086.07	R079.140.06	R079.190.10	R079.243.10
R079.087.01	R079.141.06	R079.191.10	R079.244.10
R079.087.07	R079.142.06	R079.192.10	R079.245.10
R079.088.07	R079.143.06	R079.193.10	R079.246.10
R079.089.07	R079.144.06	R079.194.10	R079.247.10
R079.090.07	R079.145.06	R079.195.23	R079.248.10
R079.091.07	R079.146.06	R079.196.10	R079.249.10
R079.092.07	R079.147.06	R079.197.10	R079.250.10
R079.093.07	R079.148.06	R079.198.10	R079.251.10
R079.094.07	R079.149.06	R079.199.10	R079.252.10
R079.095.07	R079.150.07	R079.200.10	R079.253.10
R079.096.07	R079.151.06	R079.201.10	R079.254.10
R079.097.20	R079.152.07	R079.202.10	R079.255.10
R079.099.07	R079.153.04	R079.203.10	R079.256.10
R079.100.07	R079.154.06	R079.204.10	R079.257.10
R079.101.07	R079.155.06	R079.205.10	R079.258.10
R079.102.07	R079.156.06	R079.206.10	R079.259.10
R079.103.07	R079.157.06	R079.207.10	R079.260.10
R079.104.07	R079.158.06	R079.208.10	R079.261.10
R079.105.07	R079.159.06	R079.210.16	R079.262.10
R079.106.07	R079.160.06	R079.211.20	R079.263.10

Appendix D

RO79.264.10	VL79.037.07	VL79.088.07	VL79.156.06
RO79.265.10	VL79.038.07	VL79.089.07	VL79.157.06
RO79.266.10	VL79.039.07	VL79.090.07	VL79.158.06
RO79.267.10	VL79.040.07	VL79.091.07	VL79.159.06
RO79.268.10	VL79.042.20	VL79.092.07	VL79.160.06
RO79.269.10	VL79.043.21	VL79.093.07	VL79.161.06
RO79.270.10	VL79.044.01	VL79.094.07	VL79.162.06
RO79.271.17	VL79.044.07	VL79.095.07	VL79.163.06
RO79.272.10	VL79.045.07	VL79.096.07	VL79.164.06
RO79.273.10	VL79.046.07	VL79.097.20	VL79.165.06
RO79.274.10	VL79.047.07	VL79.099.07	VL79.165.19
RO79.275.20	VL79.048.07	VL79.100.07	VL79.166.06
RO79.276.10	VL79.049.07	VL79.101.07	VL79.167.06
RO79.277.10	VL79.050.07	VL79.102.07	VL79.168.06
RO79.278.10	VL79.051.07	VL79.103.07	VL79.169.06
RO79.279.10	VL79.051.07	VL79.104.07	VL79.170.06
RO79.280.10	VL79.054.01	VL79.105.07	VL79.171.06
RO79.281.10	VL79.054.07	VL79.106.07	VL79.172.06
RO79.282.10	VL79.055.07	VL79.107.18	VL79.173.17
RO79.283.10	VL79.056.07	VL79.108.07	VL79.233.10
RO79.284.10	VL79.057.07	VL79.109.07	VL79.234.10
RO79.285.10	VL79.058.07	VL79.110.07	VL79.235.10
RO79.287.02	VL79.059.07	VL79.111.07	VL79.236.10
RO79.288.10	VL79.060.18	VL79.112.07	VL79.237.10
RO79.289.10	VL79.061.07	VL79.113.07	VL79.238.10
RO79.290.10	VL79.062.07	VL79.114.07	VL79.239.10
RO79.291.10	VL79.063.07	VL79.115.07	VL79.240.10
RO79.291.23	VL79.064.07	VL79.116.07	VL79.241.10
RO79.292.10	VL79.065.07	VL79.117.07	VL79.242.10
RO79.293.16	VL79.066.07	VL79.118.07	VL79.243.10
RO79.295.19	VL79.067.07	VL79.119.07	VL79.244.10
RO79.297.01	VL79.068.07	VL79.120.07	VL79.245.10
RO79.298.10	VL79.069.07	VL79.121.07	VL79.246.10
RO79.299.10	VL79.070.07	VL79.139.06	VL79.247.10
RO79.302.18	VL79.071.07	VL79.140.06	VL79.248.10
RO79.304.06	VL79.072.21	VL79.141.06	VL79.249.10
RO79.305.11	VL79.073.07	VL79.142.06	VL79.250.10
RO79.306.11	VL79.074.07	VL79.143.06	VL79.251.10
RO79.307.11	VL79.075.07	VL79.144.06	VL79.252.10
RO79.308.11	VL79.076.07	VL79.145.06	VL79.253.10
VL79.029.02	VL79.077.07	VL79.146.06	VL79.254.10
VL79.029.07	VL79.078.07	VL79.147.06	VL79.255.10
VL79.030.07	VL79.080.07	VL79.148.06	VL79.256.10
VL79.030.21	VL79.081.07	VL79.149.06	VL79.257.10
VL79.030.21	VL79.082.07	VL79.150.07	VL79.258.10
VL79.032.07	VL79.082.07	VL79.151.06	VL79.259.10
VL79.033.07	VL79.082.07	VL79.152.07	VL79.260.10
VL79.034.07	VL79.082.07	VL79.153.05	VL79.261.10
VL79.035.07	VL79.086.07	VL79.154.06	VL79.262.10
VL79.036.07	VL79.087.07	VL79.155.06	VL79.263.10

VL79.264.10
VL79.271.18
VL79.272.10
VL79.273.10
VL79.274.10
VL79.275.20
VL79.276.10
VL79.277.10
VL79.278.10
VL79.279.10
VL79.280.10
VL79.281.10
VL79.282.10
VL79.283.10
VL79.284.10
VL79.285.10
VL79.287.02
VL79.288.10
VL79.289.10
VL79.290.10
VL79.291.10
VY79.289.20
VY79.290.10
VY79.291.10

Appendix E

Caltech Remote Observatory Support System
Telemetry Interface Module
(Microprocessor Data Logging and Telemetry System)

TIM Specifications

CPU	IM6100 (note: 1)
Program Memory	512 words PROM 2048 words RAM
Data Storage Capacity	400 12-bit words (note: 2) with on-board battery backup
Data Inputs	
Analog	
# of Channels	8 (note: 3)
Input Impedance	5 x 10 ohms
CMRR	80 db
Input Level (Differential)	2.048 volts (full scale)
Resolution	1 mv
Digital	
# of Channels	4 (note: 3)
Type of Transmission	Serial (note: 4) 1200 baud
Voltage Levels	Space 0v. mark +5v
Sampling Interval	Variable in units of 1 minute (note: 5)
Digital Control Output	
# of Channels	16
Type of Transmission	Serial 1200 baud
Voltage Levels	Space 0v, mark +5v
Data Access	Direct-dial telephone line from central computer (note: 5)
Data Transmission Rate	1200 baud
Internal Clock	
Type	Crystal controlled
Accuracy	1 sec/day
Signal Conditioning	External (note: 6)
Phone Line	
Line Type	1 ML (line only) or 1 MB (with set)
Data Jack	RJ 45
Calif. P.U.C. Tracking Code	REQT
Ringer Equivalent	0.4B

Appendix E

Data Coupler	Universal Data Systems 1001F DAA FCC #AK396F-62376-PC-N
Power	Nominal 40 ma 10-14 volts Modem Active 500 ma 10-14 volts
Operating Temperature Range	-20 to 70 degrees C
Construction	Modular Packaged in steel case
Dimensions	12" H X 16" W x 10" D
Cost	Approx. $4,000
External Connections	Terminal strips
Required Environment:	Protection from vandalism, weather, and temperature extremes Adequate power Voice grade telephone line

Notes

1. Software compatible with PDP-8
2. Additional data memory may be added in units of 4K
3. Alternatives: 16 analog channels or 8 digital channels.
4. 7-bit ASCII plus parity (even) 12-bit data word transmitted as 2 6-bit halfwords
5. Data memory module may be removed from system without loss of contents.
6. Signal conditioning such as gain or offset adjustment will be accomplished in a satellite module close to the instrumentation. Programmable features will be controlled through the digital control output channels.

Appendix E

TIM Start Up Procedure

1. Connect power and sensor sources to appropriate terminal strip points.

2. Turn on "+12" volts.

3. Turn on "auto restart".

4. Flip sw. 8 up, 1,2,4 down, press "DIPLAY", "CTL" simultaneously, then press "RES", station ID should be displayed, "RUN" should be on, "MOD" should be off.

5. Repeat 4 with sw. 8 down. No display lights should be on. "RUN", "MOD", and "REC" should be of.

6. After 1 minute, pushing "DISPLAY" should cause station ID to be displayed and "RUN" should be on, "MOD", "REC" should be of.

This completes the initial checkout.

At this time call the Data Center to arrange for the transmission test. Data Center may not have the telephone number for the TIM station so pass this along. After arrangements have been made, hang up, plug in jack at end of cord coming from TIM cover into the telephone line as provided. TIM should then answer the incoming call from Data Center. Conversation can be monitored via the display lights (hold down "DISPLAY").

7. The data logging program can be monitored as follows: switches, 1,2,4 can be used to select an octal channel. "DISPLAY" will show the last acquired datum for the selected channel. If "CTR" is held down, the display will show the data for the selected channel continuously.

Appendix E

TIM Data Tape Format

Each file contains dump for one TIM and consists of 12 records of 768 bytes each and an end record of 512 bytes. Last file followed by two file marks.

Record 1 Format

Byte 1: 1st ID character (ASCII)
2: 2nd ID character (ASCII)
3: 0
4: Year
5: 0
6: Month
7: 0
8: Day
9-12: Origin time (GMT) in seconds
13-17: Seconds elapsed since origin
17-384: Descriptive text (ASCII)

Remaining records contain TIM data

6 data blocks/tape record

2 bytes/12 bit data word

Block Format

byte 1,2: block partition indicator (7777)

3,4: chan.# (5 bits), sampling interval-minutes (7 bits)

5,6: Max. # samples in block (6 bits)
Actual # of samples in block (6 bits)

7,8: Time of block initialization (minutes)
relative to origin time

9,10: Digital word or exponent (data related)

11, 12: checksum for block

13-128: data - unsigned octal

APPENDIX F

TECHNICAL MANUAL

CALTECH

TELEMETRY INTERFACE MODULE

Contents

INSTALLATION

Location

TIM Systems should be located in a protected environment. The case is not weatherproof.

Power

A 12 volt power source is required. This can be either a battery or a regulated power supply. TIM's will operate from a voltage of 10 to 14 volts, however the supply must be free of low frequency noise (.1 to 100HZ) of amplitude greater than .25 volts peak.

Telephone Line

Line Requirements:

Type ---------- 1MB
Ringer Equiv. --------- 0.4B
Telephone Jack -------- USOC RJ45S or RJ47S

FCC Registration Number. AK396F-62376-PC-N

Calif. PUC Tracking Code REQT

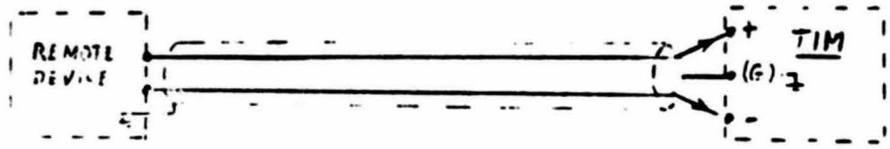
A temporary telephone connection can be made to a TIM System by attaching the two lines, "Ring" and "Tip", to the R and T terminals within the DAA mounted inside the case top. The Ring and TIP telephone lines can be identified using a voltmeter --- Ring is normally about 50 volts positive with respect to Tip.

Set plug-switch in the DAA to 0 DB.

I/O CONNECTIONS

Analog Inputs

The differential analog inputs are effective in eliminating noise pickup only if proper wiring methods are used. In electrically noisy areas all analog inputs should be connected as shown below.



Note that both the plus and the minus inputs at TIM are ungrounded.

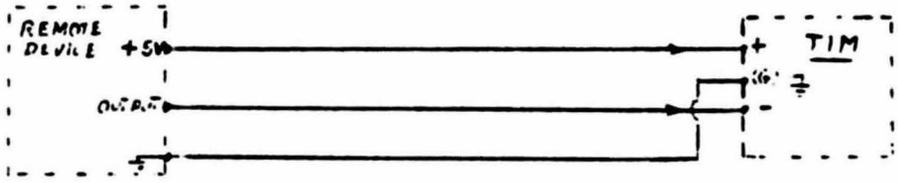
In areas with minimal electrical noise a twisted pair cable and a separate ground line can be used.

Analog inputs are ± 2.047 volts (1 millivolt per bit). Input impedance is 10 megohms. Amplifiers feeding a TIM system should have an output impedance of 2000 ohms or less. The terminal connection between + and -inputs is TIM chassis ground for attaching shields.

Digital Inputs

Digital inputs use low-power optical isolators to eliminate errors caused by transient ground-line currents or high-level noise pickup.

The recommended connecting method is shown below.



A Chassis-chassis ground line should be added to the above if it is not in for other purposes (analog shield or power ground line is adequate).

Plus 5 volts at the source equipment can be 4 to 6 volts unregulated, or + 12 can be used with a 5.6 K series resister.

The active digital input line (-) is high (zero current) when in the mark state. Mark is the normal no-data waiting level, and also logical 1.

Low inputs (ground) to a digital input port (-) cause 1 ma of current to flow from the + 5 source, thru the isolator in TIM, and back to the source equipment ground.

A digital input channel can be converted to a single-line ground referenced input by connecting the digital in + terminal to a + terminal on an empty input port (ch 12 to 15). These normally unused terminals are wired to the TIM internal + 5 supply. This defeats the optical isolation and should not be done when sending data over long lines (over 100 feet) in noisy areas. Use a heavy ground line to minimize noise problems.

Digital inputs operate at 1200 baud, 7 data bits, two stop bits. The lower 6 bits are considered to be "data" by TIM, the remaining bit is for a flag, etc. (Available for data if needed).

Digital Output

Digital output ports are compatible with the digital input ports. Optical isolation at the remote receiver is recommended in noisy areas with long lines. Output drivers are CMOS/LPTTL compatible. (40097 used for line drivers).

FRONT PANEL OPERATION

Display

The 12 bit display register on the front panel is used for monitoring and testing system operation. Information to be displayed is selected by the 3 position display select switch. The indicators are activated by pushing the DISPLAY push-botton.

The PROG position causes the program address of the next instruction fetch cycle to be set into the register whenever DISPLAY is pushed. This feature is a debugging aid not used in normal operation.

The INPUT position causes incoming channel data to be displayed. The channel to be observed is selected by the INPUT SELECT switches. The binary millivolt value for each bit is printed above the indicators. Positive voltages (sign bit off) are the sum of all bits on. Negative voltages (sign bit on) are the sum of all bits off.

Zero volts is indicated by all lamps on or all off.

The TIME position causes a display of the CPU-generated 24 hour clock. The low 6 bits are binary minutes and the upper 6 (5) bits are hours.

Also displayed are system monitoring functions.

The Run light is on when the CPU is running.

The MOD light indicates that modem power is on. This indicator should be on only when telephone communications is in progress. (Power consumption high)

The REC indicator flashes when channel input data is being recorded into the RAM data memory.

SEC goes on and off at one second intervals to indicate that the internal TCXO clock is running.

The above monitoring indicators are active only when the DISPLAY button is pushed.

Switches

The AUTO RESTART switch enables an internal 1 minute timer that is continuously cleared by the CPU during normal data aquisition. If program operation fails for any reason this timer will time out and cause a restart. The switch must be down during installation and testing and up when leaving the system for normal runs.

The CLOCK TEST switch converts the SEC indicator to a minutes indicator. It is using during testing to monitor directly (CPU not used) the accuracy of the internal TCXO. This switch should normally be down.

The INPUT SELECT switch functions are noted in the description of the display register.

RESET clears system logic, resets and restarts the CPU, and turns on modem power momentarily. Data stored in memory is not disturbed. RESET should not be pushed during normal data aquisition operation. (Channel input program stops).

CTL is a control switch sensed by the program. It is used also to clear the TCXO minute monitoring counter when CLOCK TEST is up. It is important that CLOCK TEST normally be down during data aquisiton to prevent this clear from occuring (could cause an error in the CPU 24 hour clock).

Note that display functions can be performed without interference with normal data aquisition operation. RESET pushed, or CLOCK TEST up and CTL pushed, or power off, are the panel operations to be avoided while collecting data.

SYSTEM DESCRIPTION

A block diagram of the basic TIM system is shown on the front page of the drawings located inside the rear cover of this manual.

As indicated the various cards of the system communicate via a data buss. This buss is controlled by the CPU card and contains the required data and control lines for interfacing the Intersil 6101 PIE.

Each card on the buss, except the CPU and TEST cards, contain a PIE that is attached directly to the buss. Operation of the PIE chip and the functions of the buss lines are described in the Intersil 6100 microprocessor manual.

The CPU internal data buss is attached to the peripheral data buss shown with a bidirectional buss driver. This is done to isolate the internal CPU buss from noise that may occur in the system.

The CPU card contains 2K of program RAM memory and 512 words of EPROM control/load program.

The STATUS card contains 12 bit input/output ports for monitoring and displaying system information. This card also contains system reset, restart, and TEST card interface logic, and a precision oscillator used by the CPU for generating a 24 hour system clock.

The 4K RAM card shown is for storing data readings obtained from the analog or digital input channels. The RAM card is a peripheral device on the buss as shown and appears to the CPU as identical to other cards except for device number.

The Analog Input (AIN) card contains 8 differential inputs. Channel selection and data input rates are controlled by the CPU program.

The Digital Input (DIN) and Digital Output (DOUT) cards provide 12 bit (two 6 bit half-words) digital communication with remote devices.

The Modem Control (MCTRL) cards contains the UART and power control circuits for the modem and DAA. An incomming ring signal goes directly to the Modem Control and causes power to be applied to the modem and DAA permitting telephone communications.

The TEST card shown is not supplied as part of a system. It is used only for testing and troubleshocting systems as needed. The TEST card plugs into an assigned slot adjacent to the CPU card.

For additional information refer to the individual card descriptions starting on page 10.

CARD PIE FUNCTIONS

Status Card	(Device 10)
-Write 1	Clears status input latches.
-Write 2	Sets status display register (input data, time, etc). Resets minute counter if Flag 1 is on.
-Read 1	Reads status input register.
-Read 2	Resets restart timer (data read ignored)
Sense 1	10 HZ from TCXO
Sense 2	1 HZ " "
Sense 3	1 min. " " (Minute counter clear)
RAM Card	(Device 12)
-Write 1	Set address
-Write 2	Write data
-Read 1	Read data
Analog Input	(Device 13)
-Write 1	Set channel ID (bits D9, D10, D11) and start conversion
-Read 1	Read data
Sense 1	Conversion complete (sense negative transition)
Flag 1	Selects battery monitor voltage (input channel 7)
	(Input channel 7 is normally wired internally to zero volts, switchable to battery monitor with Flag 1. Channel is available for external input if needed)
Digital Input	(Device 14)
-Write 1	Reset parity on selected channel. (All D. input channels are reset by CPU reset.)
-Write 2	Set channel ID
-Read 1	Read data. (Data is 6 low-order bits. Data bit D0 is on for parity error. Data bit D1 is high-order serial input bit - intended for high-order half-word flag when using 12 bit digital inputs)
	Digital input card PIE flag bits are available at the backplane for special control functions if needed. See Dwg.

Digital Output (Device 16)

- Write 1 Transmit data (bits D5 to D11)
- Write 2 Set channel ID (bits D8 to D11)
  Sense 1 Uart ready (sense positive level)
  Flag 1 Uart reset

Modem Control

Modem PIE (Device 17)

- Write 1 Turn off modem power (2 ms delay before off occurs)
  Sense 1 CTS from modem (Sense positive level)
  Sense 2 DRS " " " " "
  Sense 3 CD " " " " "
  Flag 1 RTS to modem
  Flag 3 DTR " "

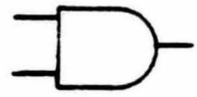
Uart PIE (Device 18)

- Write 1 Transmit data (Bits D5 to D11)
- Read 1 Read data
  Sense 1 DR from Uart
  Sense 2 TRBE from Uart
  Sense 3 TRE " "
  Sense 4 PE " "
  Flag 1 Uart reset

LOGIC SYMBOLS USED ON TIM LOGIC DRAWINGS

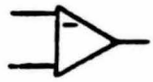
All control logic lines are labeled according to the active "go" level needed at the attached input. Data lines are labeled according to the true (1) level on the line.

Logic gate symbols are drawn conventionally as AND, OR, NAND (or -NOR) functions, however all gates are drawn to indicate their "go" condition. For example, the separate gates in a 2-input OR chip (4 gates) will be shown as a positive OR if that is the function performed, or will be shown as a negative AND if that is its active function.

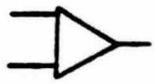


POSITIVE AND

(4081)



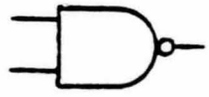
NEGATIVE OR (Negative out if any input is negative)
"Negative"=low logic level



POSITIVE OR

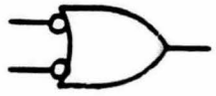
(4071)

NEGATIVE AND (Negative out if all inputs negative)



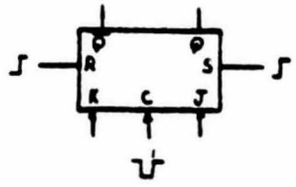
POSITIVE NAND

(4011)



NEGATIVE NOR (Positive out if any input is negative)

ONE-SHOT MV (F4528)



J-K Flip-Flop (4027)

CPU Card

The CPU card contains the Intersil 6100 CPU, a 2K Ram, 2708 EPROM, and a bidirectional data buffer connected to the TIM data buss.

(Dwg. pg. 1)

CPU memory control lines (MSEL, LXMAR, XTB) are or'd with lines from the TIM 6100 Test Unit to permit manual RAM entry and readout during trouble-shooting.

Ram and EPROM select gates are both and 'ed with a "program inhibit" off signal from the Test Unit --- this signal is activated by the Test Unit to inhibit RAM/EPROM output during a selected (program address) instruction fetch --- The Test Unit substitutes a "display accumulator"/halt sequence at the selected program address for displaying CPU data for program debugging.

EPROM-stored instructions are selected by having address bits 0, 1, & 2 on. Bits 0 to 3 are loaded into IC-13 with LXMAR --- IC-14 is a negative AND gate that checks for low levels on the $\bar{Q}$ side of the IC-13 latches. A low at IC-14/6 is and'ed with -MSEL and $\bullet\overline{PI}$ at IC-17/15, giving a negative transition that triggers IC-12/6, and a -WAIT level that stops the CPU to provide time for packing two 6 bit EPROM characters into a normal 12 bit instruction format.

EPROM-stored instructions are fetched as follows:

1. IC-12/7 goes low which sets a low level on the low-order EPROM address bit (IC-18 pin 8), selecting the first two (most significant) octal characters of the instruction. These appear on lines R5 thru R0.
2. The above low address select level returns to a high state which strobes the above characters into IC-5 & 7 latches.
3. The low-to-high level change above also selects the next sequential odd address in the EPROM. The second two (lowest) octal characters of the instruction are gated to the DX data buss by -MSEL.CE, which has also gated the output of the IC5 & 7 latches to the buss.
4. IC-12/6 provides a negative transition that triggers the "continue" one-shot IC-12/10 simultaneously with the positive transition of IC-12/7. This causes the -WAIT signal to go off (high) via IC-17 pin 5, which in turn causes the CPU to continue the instruction fetch cycle.

   The above EPROM instruction fetch cycle utilizes two inherent delays. The "slow" access time of the EPROM holds the output of the first 6 bits stable when the high transition of IC-12/7 sets the latches and selects a higher EPROM address. Also the CPU start-up delay after a WAIT allows the second 6 bits to stabilize on the DX lines before the CPU instruction input transfer.

NOTE: Gates are designated by output pin. IC-14/6 indicates gate that has pin 6 for output, etc.

The diodes on CPU pin 11 and the parallel inputs to IC 17/15 and 17/1 are used to minimize the delay between MSEL and -WAIT.

The Output Buffers, IC 1 & 2, normally transfer CPU buss data to the TIM data buss. During a device input to the CPU, the Output buffers are disabled (3 state) and the Input Buffers IC 3 & 4 are enabled.

The CPU data buss is connected to the CPU RAM address buss via IC 22 & 23 address drivers. The RAM is selected if the address bit zero is off. Ram selection is by -CS2 from IC 13 at the top of page 1. -CS1A/CS1B select the lower or upper K of RAM.

Minimal logic is used for EPROM/RAM selection. Proper operation requires that 0-2K be treated as RAM, and 3.5-4K be EPROM.

The program wait latch on page 2 can be wired at the backplane to cause a CPU wait as needed to conserve power. This latch is not used in the standard version of TIM.

STATUS CARD

The Status Card contains the following:

1. A 12 bit input port for reading hardware status information.
2. A 12 bit output port for displaying data as determined by the program, and for displaying program address.
3. A TCXO clock used as a reference by the CPU for implementing a programmed 24 hour clock.
4. System support logic --- including CPU (system) reset control and automatic restart timing.

(page 1)

ICs 10, 11, & 12 at the upper left drive the front panel display LEDS. Data to be displayed is stored in the ICs 6, 7, & 8 latches, which are set by the program (Status write 2), or set during an instruction fetch cycle for program address display.

Programmed display of channel input data is enabled by a low level at IC 24/10 and 27/6.

Display of the 24 hour clock is done via the same logic --- the program senses the position of the front panel display select switch at status card pin 67 at the right of the page.

Program address display is activated by a low at IC24/11, which is inverted thru IC 16/13 for a high at IC 17/4. LXMAR strobes during a fetch cycle strobe the instruction address into the display latches via ICs 16/4, 16/3, 17/6, and 25/3.

The address display described above is intended to provide minimal monitoring capability for isolating problems without the Test System installed. A current address is frozen into the display whenever the DISPLAY button is pushed --- this is achieved by the IC 18 latches which inhibit further address loading when the DISPLAY button is held down.

When the Test System is installed, a low at IC 17-15 inhibits the freeze latches, permitting continuous active address display for step mode operation, etc.

Note that when the test card is installed all data/instructions/addresses are displayed on the Test System panel. The DISPLAY button should not be used.

The display drivers shown are wired-or at the output (backplane) with additional display drivers in the Test Card.

When the Test Card is installed, the TST+ input at the left holds power on the above display drivers and disables the display select switch. Displays are selected on the test panel.

The status input port, IC 3 & 4 at the right contains 8 direct level input lines and 4 latched (IC 2) lines. In TIM systems, the three latched input shown also cause a CPU Reset --- the program then interprets these latches to determine which line caused the reset.

The above latches are reset by a -write 1 as shown.

IC-14 is for generating a CPU wait during device-to-CPU data transfers. DEVSEL causes card pin 14 to go low, placing the CPU in a wait state. DEVSEL also triggers IC-14/6, which times out and triggers IC-14/10, which causes -wait to off (high).

(page 2)

IC-9 in the upper left corner drives the system status indicators, RUN, MOD, and REC, and the SEC indicator used for monitoring the TCXO clock.

One-shots IC-22 and 23 generate the reset cycle. Reset triggers are or'd at IC-19/15. The leading edge of IC-22/6 output stops the CPU via IC-20/13, IC-19/1, and IC-15/9. The trailing edge triggers IC-22/10, which generates the -reset pulse. The trailing edge of the reset pulse triggers IC-23/6, which provides a short delay prior to restarting the CPU via IC-23/9, 19/1, and IC-15/9.

IC-20/3 and 20/4 cause the CPU to stop following a fetch cycle when operating in step mode with the test card. The -T R/H line at IC-19/2 is the step mode start signal from the test card.

The TCXO clock is located in the upper-right corner of the drawing. The 4518 IC's are BCD dividers to generate seconds and minutes pulses from the 10 khz TCXO.

The TCXO can be adjusted to improve time accuracy if necessary.

Standard versions of TIM use the seconds pulse for generating 24 hour time. IC-32/3 and 32/4 permit a programmed reset of the minutes counter for synchronizing the minutes pulse with real time if needed.

IC-30 is the restart counter, which generates a restart pulse via IC29/9 one minute after failing to receive "continue" pulses (-read 2) from the program.

The value of this feature depends upon the applications and program --- in TIM system systems with remote programming capability (RAM stored program), restart attempts are noted in system status information returned with data via TP. Since a prom program is not available for immediate reloading, restart does not actually occur.

Restart timeout is inhibited if the AUTO RESTART switch on the front panel is down. Restart-off is needed for testing, step mode, etc.

Operation of the TCXO can be checked without a program running or the test system by the logic consisting of IC8s 28/1, 26/11, etc. If the front panel CLOCK TEST switch is up, the seconds indicator is converted to a minutes indicator -- the lamp compliments each minute. Synchronizing with a time reference can be done by pushing CTL at an exact minute (minutes counter reset).

RAM Card

Ram address information is set into IC7,8 & 9 latches by Write 1 from the PIE. Write 1 is also transmitted through IC2/4 to the one shot IC14/10 on page 2. The output pulse from IC14/10 disables the RAM select lines at IC13/1, enables the 3-state outputs of IC7, 8, &9, and strobes (trailing edge) the address into all RAM chips. The two high order bits are strobed into IC12 as shown on page 2.

Data is written into RAM using a similar technique. The RAM bank (K) selection by IC13 is not disabled during write.

Read is accomplished by read 1 from the PIE which enables the 3-state drivers IC4,5,&6, which gate the RAM output to the TIM data buss.

The reset input at IC15/13 on page 2 is for preventing write strobes during system reset and power on. Additional protection is provided by IC15/3, which holds CS2 high (off) for a short time after power on occurs.

Transistors T1and T2 feed TIM +5 power to the RAM chips and control charging of the backup battery.

Address information from the Auxillary I/O port is strobed into RAM by the -A Set Addr line. Write data is strobed in by -A Write. Read data is always on the memory buss except when setting address or writing.

The -CPU Set Addr line and the R+W line are used by the external device to make sure that the CPU is not currently executing a set addres -- read or write operation. The External device should use the IFETCH line available at the CPU connector for determining when the RAM can be accessed in addition to the above lines. Access should begin at the beginning of IFETCH.

Analog Input Card

IC4/11 & 12 generate a 12 volt square wave that is fed into the diode-capacitor voltage inverter circuit attached to pin 7 of IC5. Negative 10 volts exists at this point which is connected to the converter and Mux through a noise filter.

IC9 operates as a unity gain amplifier for converting the differential input lines to a single-line ground referenced signal at pin 5 of the A/D converter.

Channel ID is set into the IC7 latches via a one-shot by Write 1 from the PIE. ID is set by the leading edge of the pulse and the converter is started by the trailing edge.

The amplifier zero adjustment is the black pot adjacent to the amplifier. This should be adjusted with both input lines grounded on the selected channel.

Full-scale range can be adjusted with the green pot located at the upper outside corner of the card. This adjusts the reference voltage applied to the converter.

Input Channel 7 is normally jumpered on the card to a voltage divider for measuring battery voltage (12 volt power input). This channel is available for external input if needed.

Digital Input Card

Serial input data is sent to the input UARTS IC 15 thru 18 via the optical isolators, IC11-14. The + input pins at the card connector are connected to +5 at the isolated device that is sending data to TIM.

When a UART receives a data word (7 bits), pin 19 of the UART goes high triggering the one-shot which activates a PIE sense input.

The TIM CPU responds by setting channel ID into IC2 with a PIE write 2. The selected channel is then read with read 1.

IC9 selects the UART parity output for the selected channel and this is read in as data bit 0. Parity for the selected channel only can be reset with write 1. CPU reset from the TIM buss resets all channels.

Digital Output

The output channel is set into the IC3 latch by write 2.

Data is transferred to the UART by write 1.

The UART is reset by setting flag 1 on and then off.

The UART is available for a new transmission when sense 1 is a high level.

Modem Control

The Modem Control card contains the power conversion circuits for the TIM system and the modem interface logic.

The +5 buss power for TIM is generated by the LM309 regulator shown at the upper left of the drawing. ± 12 for the modem, DAA, and the CPU 2708 EPROM (+12,-5) is generated by the DC/DC converter.

Plus 12 is applied to the converter when IC9 pin 2 is low. This is the Q side of the latch, which is turned on by CPU reset (EPROM needs pwr.)

The above power-on latch is turned of by write 1 from the modem control PIE, IC1.

IC9/15 is set by a positive transition on the +12 battery input line, to detect power failures. The output of this latch is normally wired to an input line on the TIM Status card.

Modem and EPROM power is normally off when the system is aquiring data. When an incomming call occurs, The DAA ring signal at card pin 8 is integrated to eliminate noise problems and triggers IC11/6.

IC11/6 triggers IC11/9 to prevent a second ring from again triggering IC11/6 (IC11 pin 3 low clears the one-shot).

The leading edge of the pulse at IC11/6 activates the PIE sense input which signals the program that a ring has occured. The program should complete its immediate operation and then stop.

The trailing edge of the IC11/6 pulse triggers IC7/9 which causes a system reset at the Status Card and also sets the ring flag latch input on the Status Card. Note that ring inputs force a hardware restart independent of the program. This is to prevent program RAM (power) failures from locking out the system from telephone communications.

IC3 is the modem UART control PIE. Data is sent to the UART by write 1 from the PIE.

Data received is indicated by the DR line wired to pin 7 (sense 1) of the PIE. Data is read from the UART to the TIM buss by read 1.

The UART is reset by setting flag 1 on and off.

Parity error is indicated by a high level at IC3/4 (sense 4)

TRE and TRBE are UART transmit busy logic lines.

MAINTENANCE ACCESS

Card Rack Removal

1. Disconnect 12 volt power source.
2. Remove 4 screws at outside rim of front panel.
3. Push panel slightly to the left and lift. Set rack momentarily on top of case rim (rotate rack about 20°).
4. Disconnect plugs at lower right of rack.
5. Remove rack.

Card Access

1. Remove 4 larger screws from the front panel.
2. Lift panel off the rack and set aside at left.
3. Remove cards as required.

Card Extender Use

Trouble-shooting using the card extender is easier if the rack is laid on the short side with the I/O connectors (step 4 above) up. Lay front panel to the right.

Test System

With the rack in the position recommended above, the test card can be plugged into its PC connector located at the right end of the rack.

Data Coupler

The Data Coupler (DAA) case is permanently attached to the lid of the TIM case. The cover can be pulled off to obtain access. The Coupler electronic board can be removed by removing the 4 screws at the corners of the board.

Telephone communications while the system is disassembled as above can be accomplished by plugging in a cable extender from the Coupler cable to the front panel. If a cable is not available, the Coupler board and cable can be removed and positioned near the front panel (fasten the Coupler cable clamp to the board temporarily with a 4-40 nut to protect connections).

Back Plane Access

Remove 4 screws from the back plane and remove cover.

CAUTION

The CMOS logic in TIM can be damaged by static discharges or other high voltages. Make sure all instruments are grounded to the rack before attaching probes. If possible all equipment should be attached to earth ground. In areas with low humidity touch the rack frame before starting work or use a wrist ground strap.

NOTE To remove RAM for data recovery, turn off power and remove RAM card as indicated above. Place card in an insulated bag. If possible return entire system (do not disturb RAM) for data readout.

PARTS LIST

Digital IC's

The following IC numbers correspond to types provided by both RCA and Fairchild, except where the number is preceeded by CD (RCA only) or F (Fairchild only). These exceptions represent availability when the list below was made.

IC Type	No. Req'd/System
4011	8
4023	3
4012	1
4081	2
4071	2
4072	1
4075	2
4027	3
4044	1
4076	8
CD4514	1
4518	4
F4523	13
F4539	1
CD4556	2
40097	22
40098	9
40175	6
F4702	1

Misc. IC's

		MFG.	Function
74LS38N	4	TI	OC Nand gate
6N139	4	HP	Optical iso.
2708	1	TI	EPROM
643-1	1	Teledyne	S/S relay

Modules

C0236T-1	1	Vectron	Clock Osc.
30C12-12D125	1	Semiconductor Cir.	Dc/DC Conv.

Analog IC's

AD7550	1	Analog Devices	A/D Conv.
HI1-507-2	1	Harris	Analog Mux
AD522A	1	Analog Devices	Inst. Amp.

Regulators	P/N	Quan.	MFG.	Function
	REF02J	1	PMI	A/D Conv. ref.
	7905	1	TI	-5 volt reg.
	LM309	1	Mot.	+5 volt reg.
	RC4194D	1	Raytheon	±6 volt reg.
Transistors				
	2N697	1	TI	(MCTRL)
	2N3638	1	Fairchild	(RAM)
	2N2222	1	Raytheon	(RAM)
Diodes				
	1N748	1	TI	(RAM)
	1N4735	2		5 V. zener
	1N3998R	1	Mot.	5 V. OVP
	1N270	11		Ge. switching diode
	1N3311	1	Mot.	12 V. OVP, Rev. V.
Intersil CMOS				
	IM6100IDL	1		CPU
	IM6101IPL	7		PIE
	IM6402IPL	4		UART
	IM6403IPL	2		UART
	IM6518IJN	72		RAM
Potentiometers				
	1240X	1	Vishay	5K 10ppm (AIN)
	E4A10?	1	Allen-Bradley	10K
Crystals				
	2MHZ	1	American Crystal Labs	All 20pf parallel resonant
	307.2KHZ	2		
	2.4576MHZ	1		
Capacitors				
	SXM-250	1	Mallory	.005MFD for A/D
	2140-Z5U-50R-474K	1	Veradyne	.47MFD Tantalum
	2140-Z5U-50R-105Z	44	"	1MFD "
	TE1127	8	Sprague	5 MFD, 12 V.
	TE1204	1	"	10 " 25 "
	TE1131	3	"	25 " 12 "
	TE1133	3	"	50 " 12 "
	TE1140	2	"	390 " 12 "
Connectors				
	H321150	10	TI	100 pin card conn.
	57-40360	1	Amphenol	36 pin socket
	57-30360	1	"	" " plug
	57-40240	1	"	24 pin socket

Connectors (Cont'd) P/N	Quan.	MFG.	Function
57-30240	1	Amphenol	24 pin plug
125-221	1	"	9 pin socket
126-220	1	"	9 pin plug

Front Panel Parts

MV5020	16	Monsanto	LED
MRP2-53	1	Alco	2 pole 5 pos. sw.
MT-123	4	"	SPDT toggle sw.
T-223	3	"	DPDT " "
10-3-13	1	Raytheon	Knob
342004	1	Bussman	Fuseholder

Terminal Strips

[illegible]CX-2	1	Curtiss	Power Input
SEIPCX-12	4	"	Signal Inputs
SEIPCX-16	2	"	Control Outputs

RAM Battery

GCK15OST	3	GE	Backup Batt.

TIM TEST SYSTEM OPERATION

OPERATION		DISPLAY SW.	FUNCTION SW.	NOTE
Normal Run		not MEM	RUN	RESET to start. Push STEP for stop/start.
Address Stop		ADDR or INST	ADDR STP	Set addr in sw 0-11 before starting. To continue running switch to RUN and push STEP. To enter Step Mode switch to STEP and push STEP P.B.
Step Mode		INST, ADDR, or REG	STEP	Push STEP P.B.
Display A		REG	DSPA	Set addr in sw 0-11 before starting. To restart push RESET.
RAM Load		ADDR	LOAD	(1) Set addr in sw 0-11 and push LDAD. (2) Set data in sw 0-11 and push LDMEM. To enter data into next location push COUNT and repeat (2)
RAM Read	(1)	ADDR	LOAD	(1) Same as above
	(2)	MEM	STEP (not LOAD)	Data is displayed in indicators. Push COUNT to examine next location.

NOTE: (1) The Display Switch MEM position, and the Function Switch LOAD position disables normal memory gating --- the CPU will not run if these settings exist.

(2) The Instruction Display buffer is always set during a fetch cycle. The Instruction Address Display buffer is set only if the Display switch is not in the REG position. (ADDR and REG use the same buffer)

(3) The TIM front panel data display should not be used when the test system is plugged in.

```
'      DIGITAL ENGINEERING CO.
'
'      W/L:  BACKPLANE  (Buss wiring)
'
'BOARDTYPE   BACKPLANE                          WIRE: 30 AWG
'
'ZERO STAK1  C1002                              STRIP: 1"
'
'REMARKS    ** WIRE JUMPERS FIRST **
'
 SIG
C0170 C0270 C0370 C0470 C0570 C0670 C0770 C0870 C0970 !XC0170
C0171 C0271 C0371 C0471 C0571 C0671 C0771 C0871 C0971 !XC0171
C0175 C0275 C0375 C0475 C0575 C0675 C0775 C0875 C0975 !XC0175
C0176 C0276 C0376 C0476 C0576 C0676 C0776 C0876 C0976 !XC0176
C0177 C0277 C0377 C0477 C0577 C0677 C0777 C0877 C0977 !XC0177
C0178 C0278 C0378 C0478 C0578 C0678 C0778 C0878 C0978 !XC0178
C0181 C0281 C0381 C0481 C0581 C0681 C0781 C0881 C0981 !XC0181
C0182 C0282 C0382 C0482 C0582 C0682 C0782 C0882 C0982 !XC0182
C0183 C0283 C0383 C0483 C0583 C0683 C0783 C0883 C0983 !XC0183
C0184 C0284 C0384 C0484 C0584 C0684 C0784 C0884 C0984 !XC0184
C0185 C0285 C0385 C0485 C0585 C0685 C0785 C0885 C0985 !XC0185
C0186 C0286 C0386 C0486 C0586 C0686 C0786 C0886 C0986 !XC0186
C0187 C0287 C0387 C0487 C0587 C0687 C0787 C0887 C0987 !XC0187
C0188 C0288 C0388 C0488 C0588 C0688 C0788 C0888 C0988 !XC0188
C0189 C0289 C0389 C0489 C0589 C0689 C0789 C0889 C0989 !XC0189
C0190 C0290 C0390 C0490 C0590 C0690 C0790 C0890 C0990 !XC0190
C0191 C0291 C0391 C0491 C0591 C0691 C0791 C0891 C0991 !XC0191
C0192 C0292 C0392 C0492 C0592 C0692 C0792 C0892 C0992 !XC0192
C0193 C0293 C0393 C0493 C0593 C0693 C0793 C0893 C0993 !XC0193
C0194 C0294 C0394 C0494 C0594 C0694 C0794 C0894 C0994 !XC0194
```

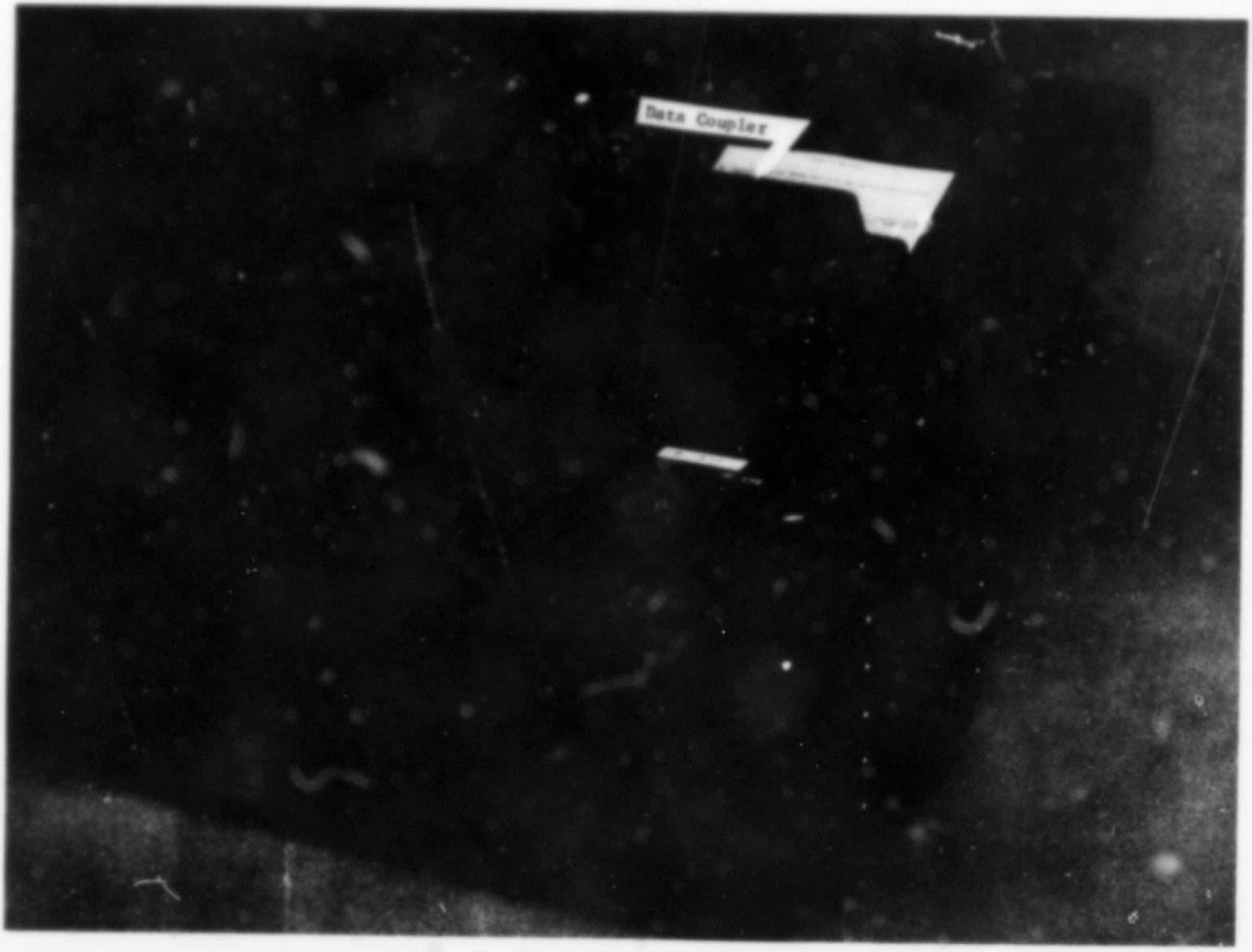
NOTE: BACKPLANE WIRING IS INDICATED ON THE LOGIC DRAWINGS.
All front panel switch/indicator wiring is shown by dashed lines at the card connectors. Wiring to other cards is noted by a card number (C2,C3, etc.). Refer to the dwg. for the card indicated and locate the identical signal label to find the pin number on the second card.

```
'      DIGITAL ENGINEERING CO.
'
'      W/L:  BACKPLANE
'
'BOARDTYPE   BACKPLANE                          WIRE  30 AWG
'
'ZERO STHM1   C1002                             STRIP: 1"
'
'REMARKS   ** WIRE JUMPERS FIRST **
'
 SIG
C0142 C0245 C0210 'XC0142
C0144 C0249 'XC0144
C0156 C0271 'XC0156
C0165 C0247 'XC0165
C0145 C0232 'XC0145
C0143 C0226 C0166 'XC0143
C0167 C0267 'XC0167
C0158 C0245 'XC0158
C0155 C0243 'XC0155
C0169 C0264 'XC0169
C0172 C0245 'XC0172
C0195 C0271 'XC0195
C0196 C0249 'XC0196
C0197 C0216 'XC0197
C0162 C0261 C0206 'XC0162
C0297 C0248 'XC0297
C0295 C0246 'XC0295
C0121 C0242 'XC0121
C0122 C0238 'XC0122
C0123 C0236 'XC0123
C0124 C0234 'XC0124
C0125 C0232 'XC0125
C0126 C0230 'XC0126
C0127 C0241 'XC0127
C0128 C0237 'XC0128
C0129 C0235 'XC0129
C0130 C0233 'XC0130
C0131 C0231 'XC0131
C0132 C0279 'XC0132
C0100 C0473 'XC0273
C0374 C0473 'XC0374
C0474 C0573 'XC0474
C0574 C0673 'XC0574
C0674 C0773 'XC0674
C0774 C0973 'XC0774
C0974 C0873 'XC0873
C0102 C0170 'XC0102
C0101 C0119 'XC0101
C0140 C0160 'XC0140
C0139 C0159 'XC0139
C0940 C0960 'XC0940
C0939 C0959 'XC0939
C0201 C0270 C0232 C0237 'XC0201
C0258 C0370 'XC0320
C0354 C0317 'XC0317
C0135 C0356 'XC0135
C0257 C0344 'XC0257
C0263 C0246 C0307 'XC0263
C0253 C0100 'XC0253
C0308 C0375 'XC0308
C0312 C0381 'XC0312
```

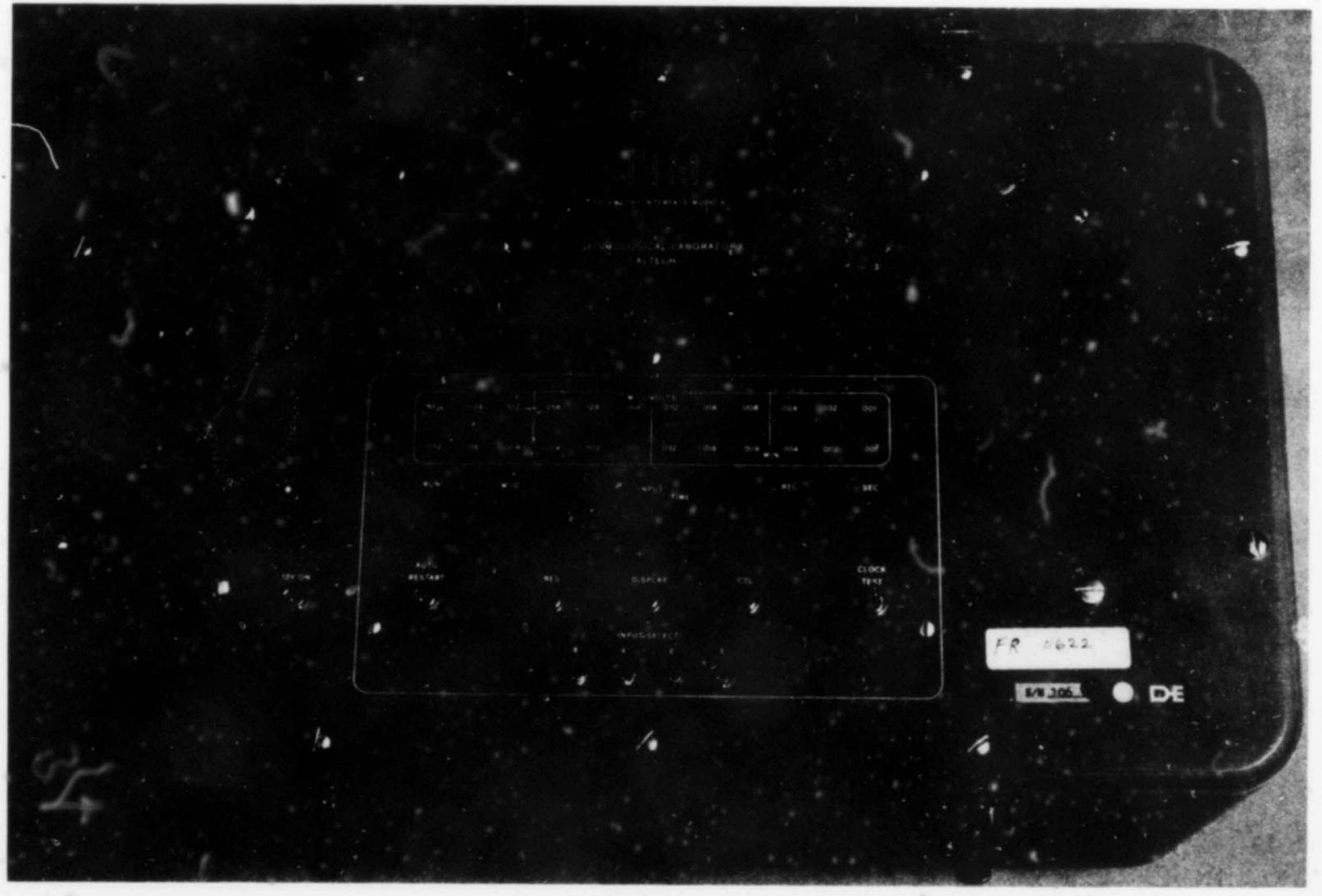
(C0142= Card 1,pin 42, etc.)

(Ignore XC entries)

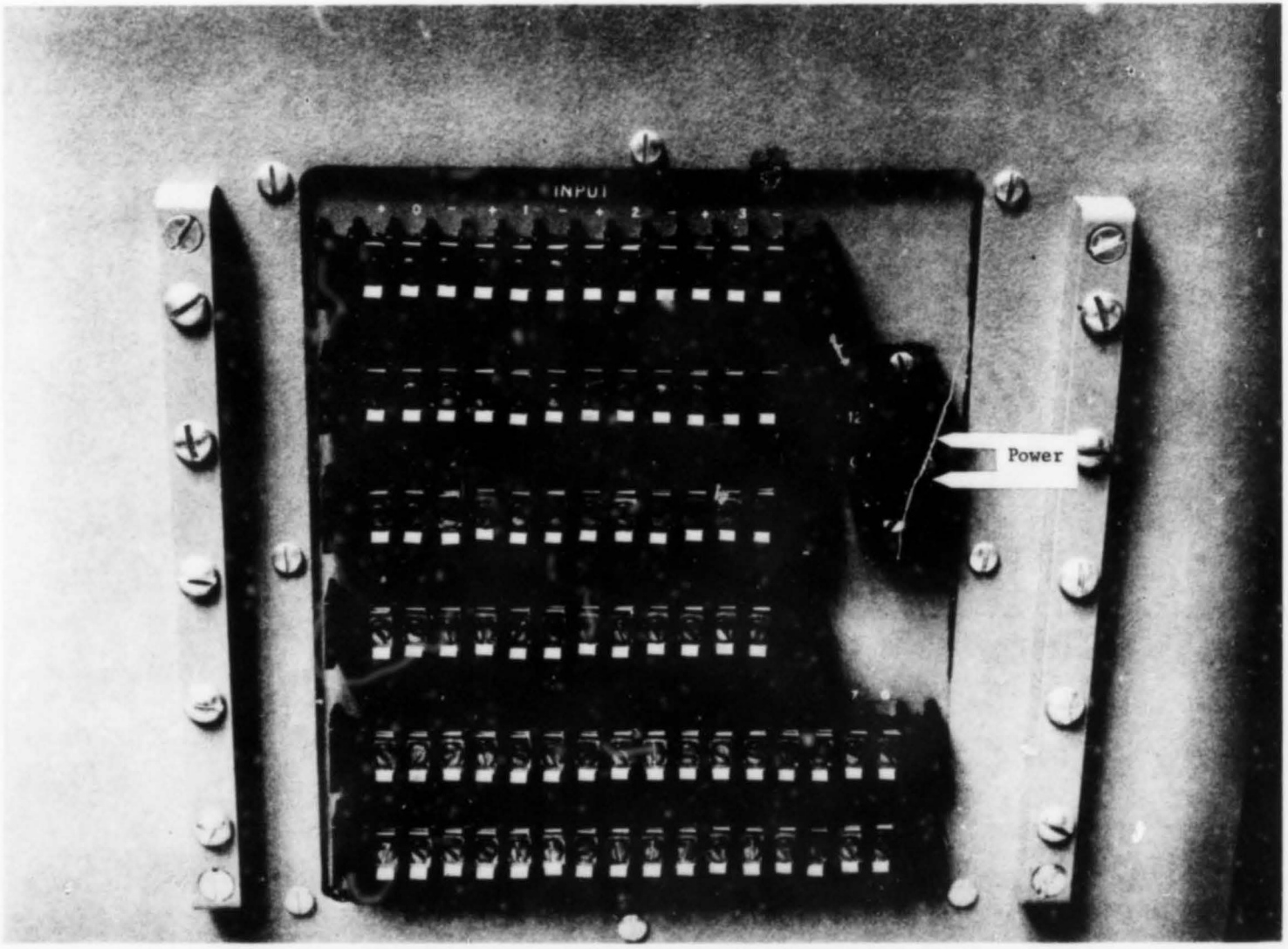
```
C0328 C0[illegible]4 !XC0328
C0323 C0348 !XC0323
C0365 C0[illegible]13 !XC0313
C0362 C03[illegible]4 C09X2 !XC0324
C0369 C0917 !XC0369
C0345 C09[illegible]8 !XC0345
C0352 C0[illegible]8 !XC0352
C0166 C0164 C0163 C0199 !XC0463
C0938 C1042 !XC0938
C0942 C1006 !XC0942
C0968 C1046 !XC0968
C0966 C1018 !XC0966
C0964 C1041 !XC0964
C0996 C1040 !XC0996
C0995 C1048 !XC0995
C1012 C1072 C1032 C1040 !XC1012
C1083 C1084 C1085 !XC1083
C1087 C1088 C1089 !XC1087
C1003 C1005 C1024 !XC1003
C09100 C[illegible]30 C1024 !XC0980
C0999 C0979 C1024 !XC0979
C0947 C0948 C1087 !XC0947
C0955 C09[illegible]6 !XC0955
C0951 C09[illegible]2 C1083 !XC0951
C0945 C0946 !XC0945
C0902 C0970 C1012 !XC0902
C0901 C0919 C1022 !XC0901
C0298 C09[illegible]1 !XC0298
C0296 C09[illegible] !XC0296
```



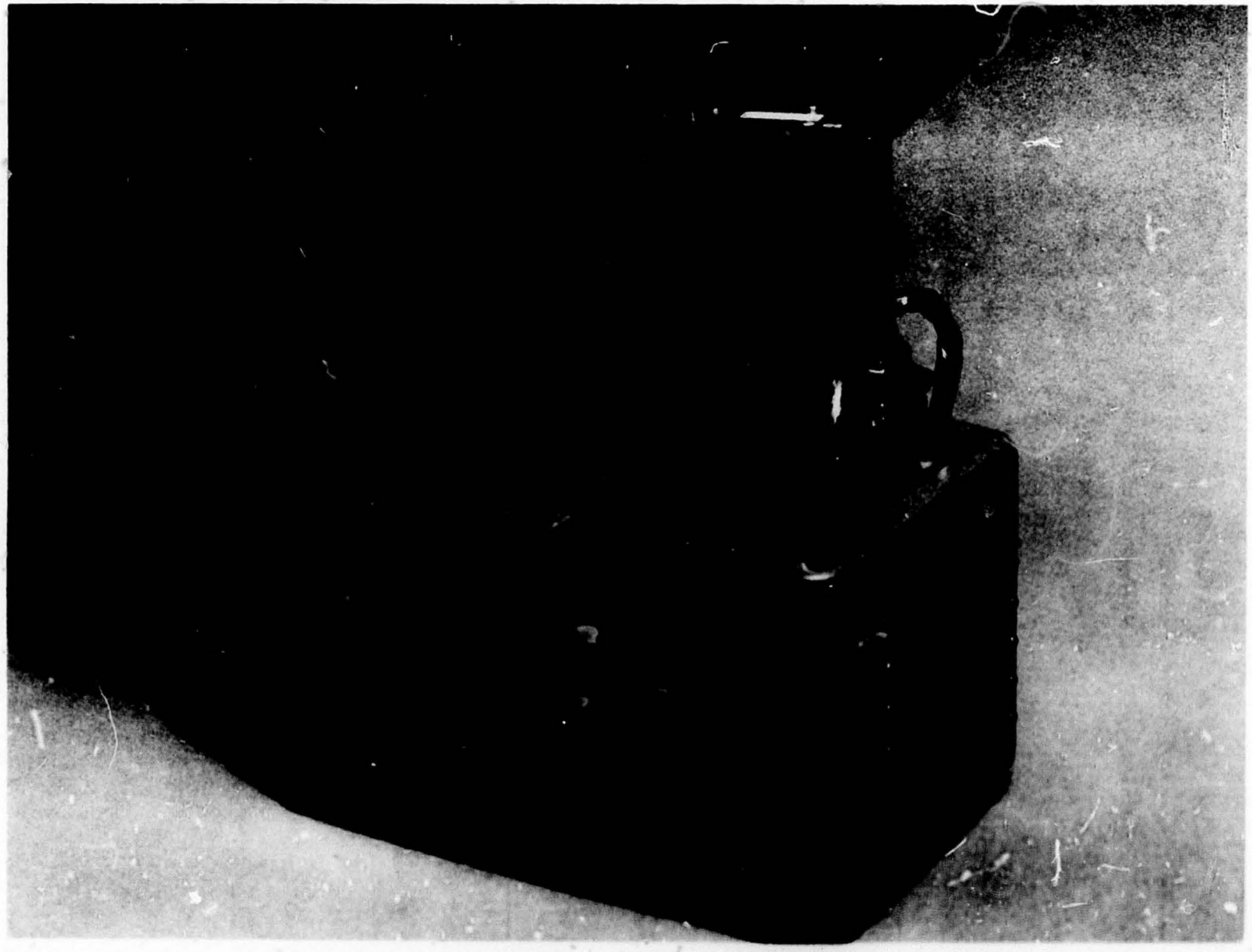
1. Photograph of a TIM unit.



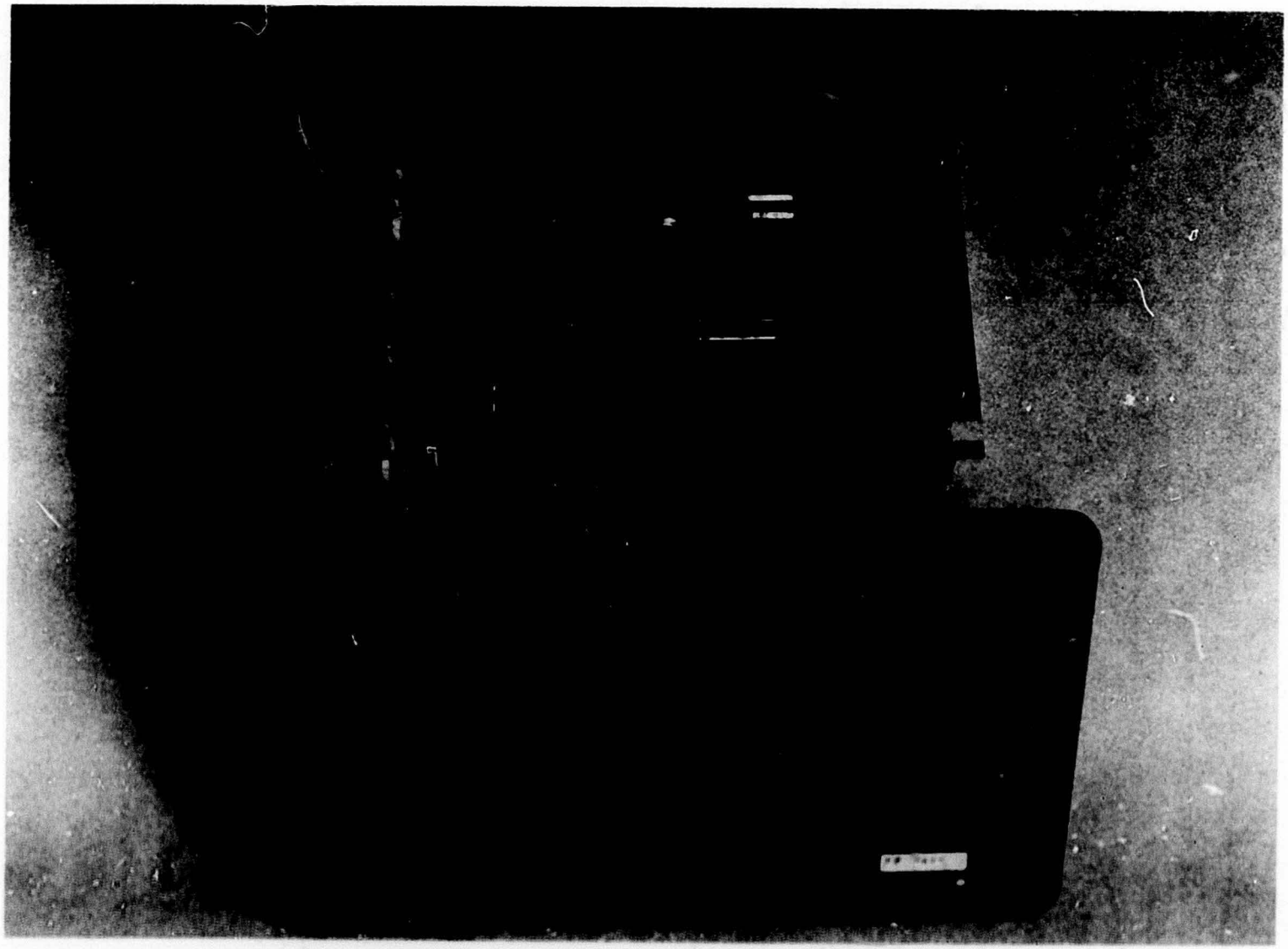
2. Front panel of TIM unit.



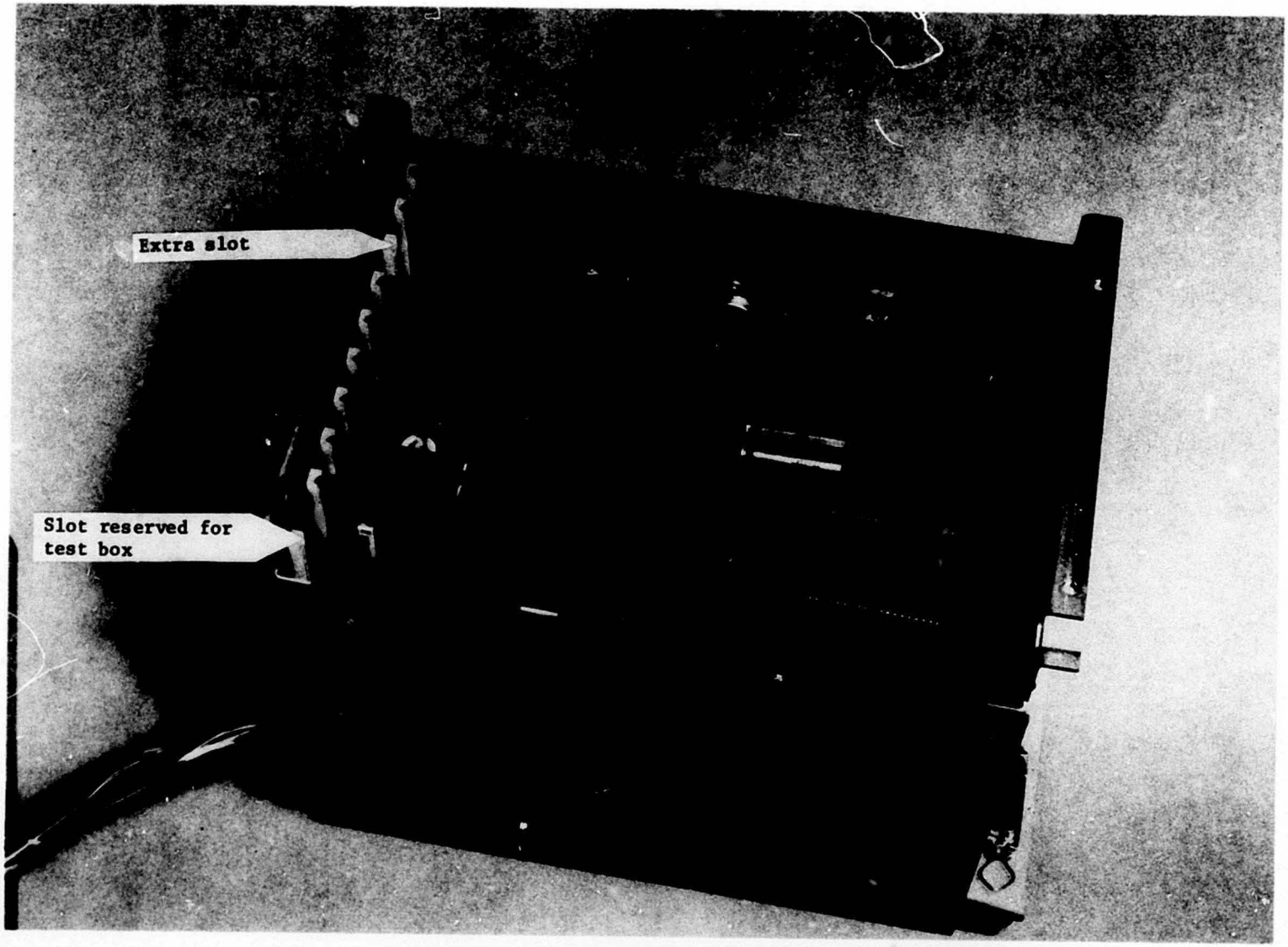
3. Closeup of input/output terminal panel.



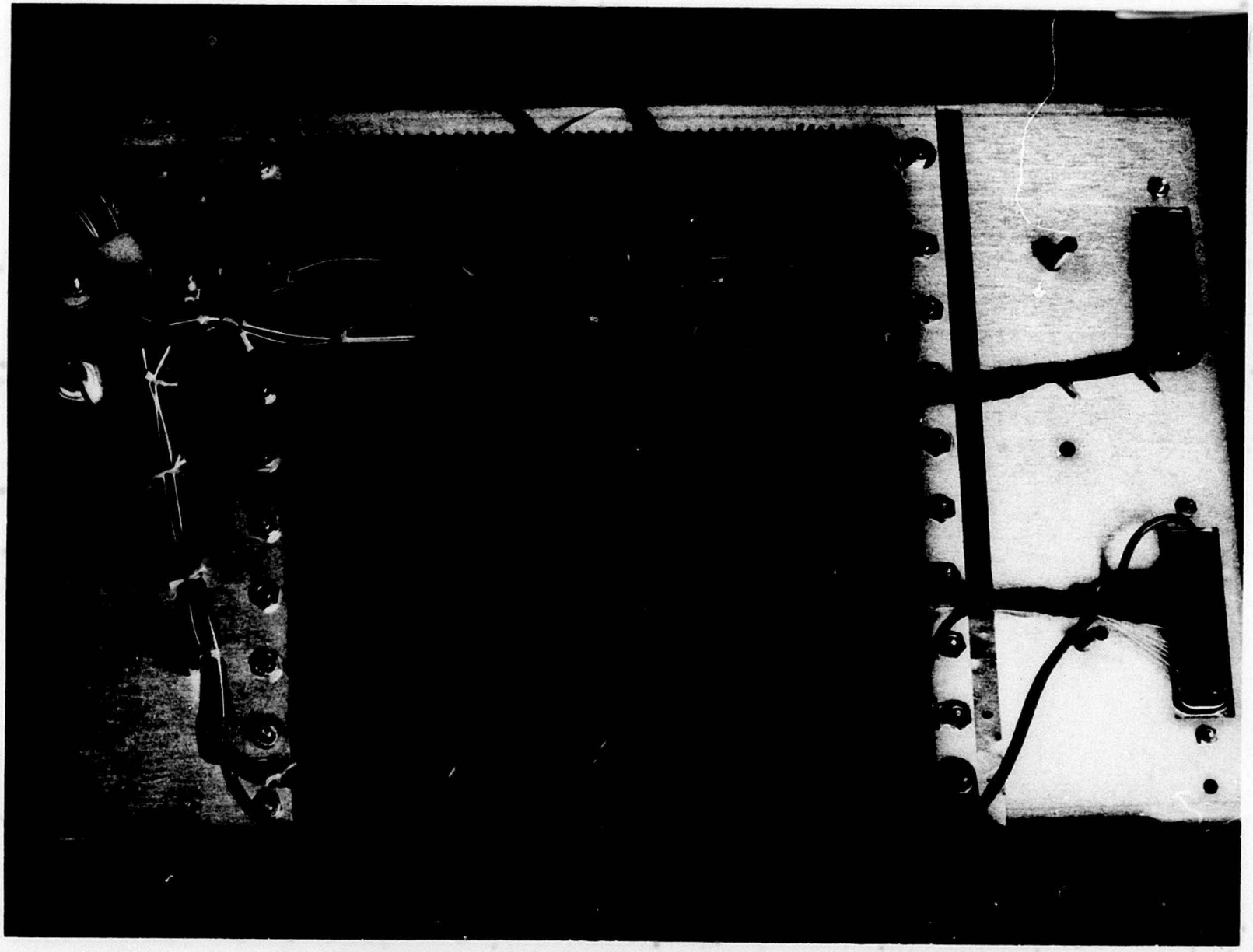
4. TIM unit resting on its case.



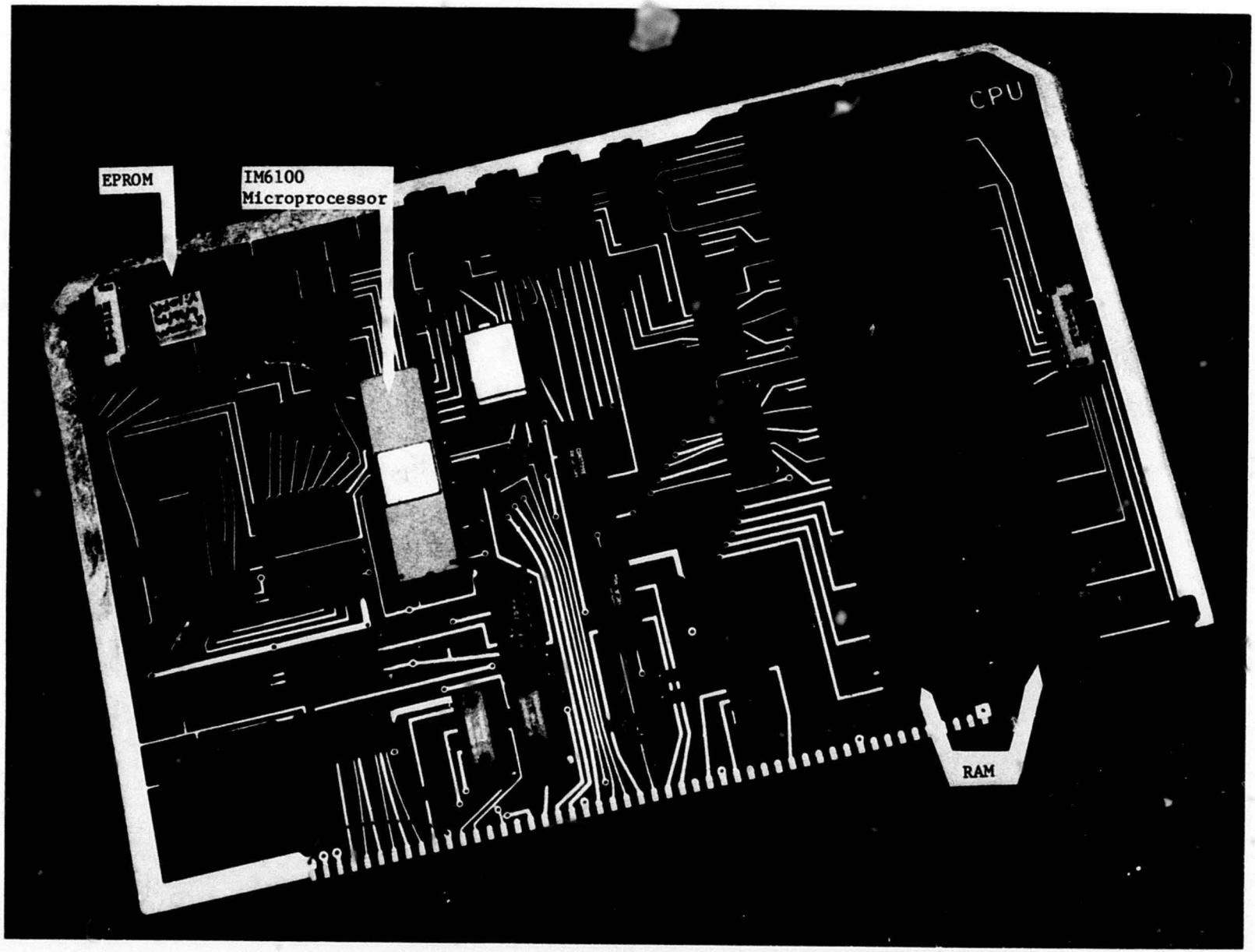
5. Front panel unbolted and removed from card cage.



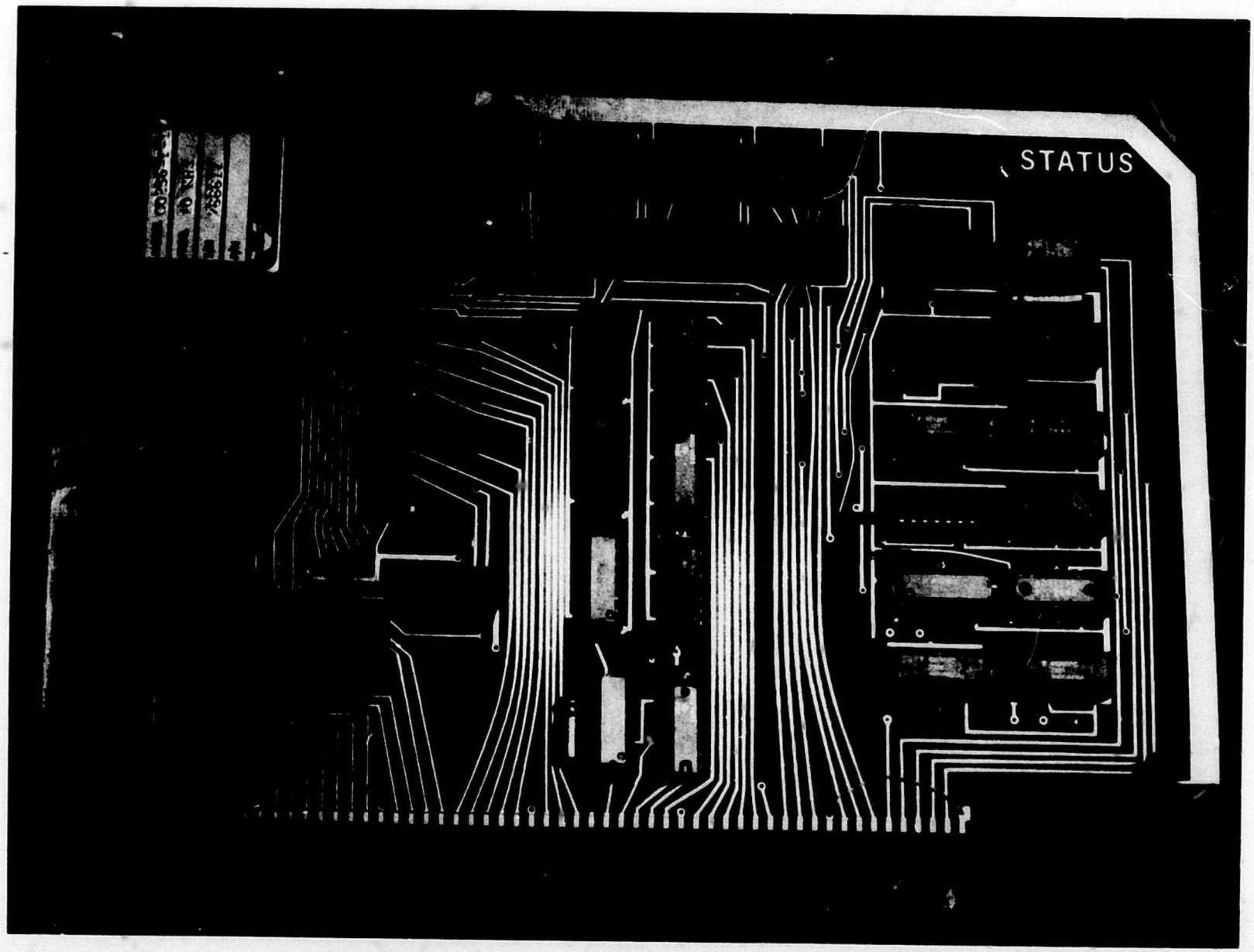
6. Closeup of card cage.



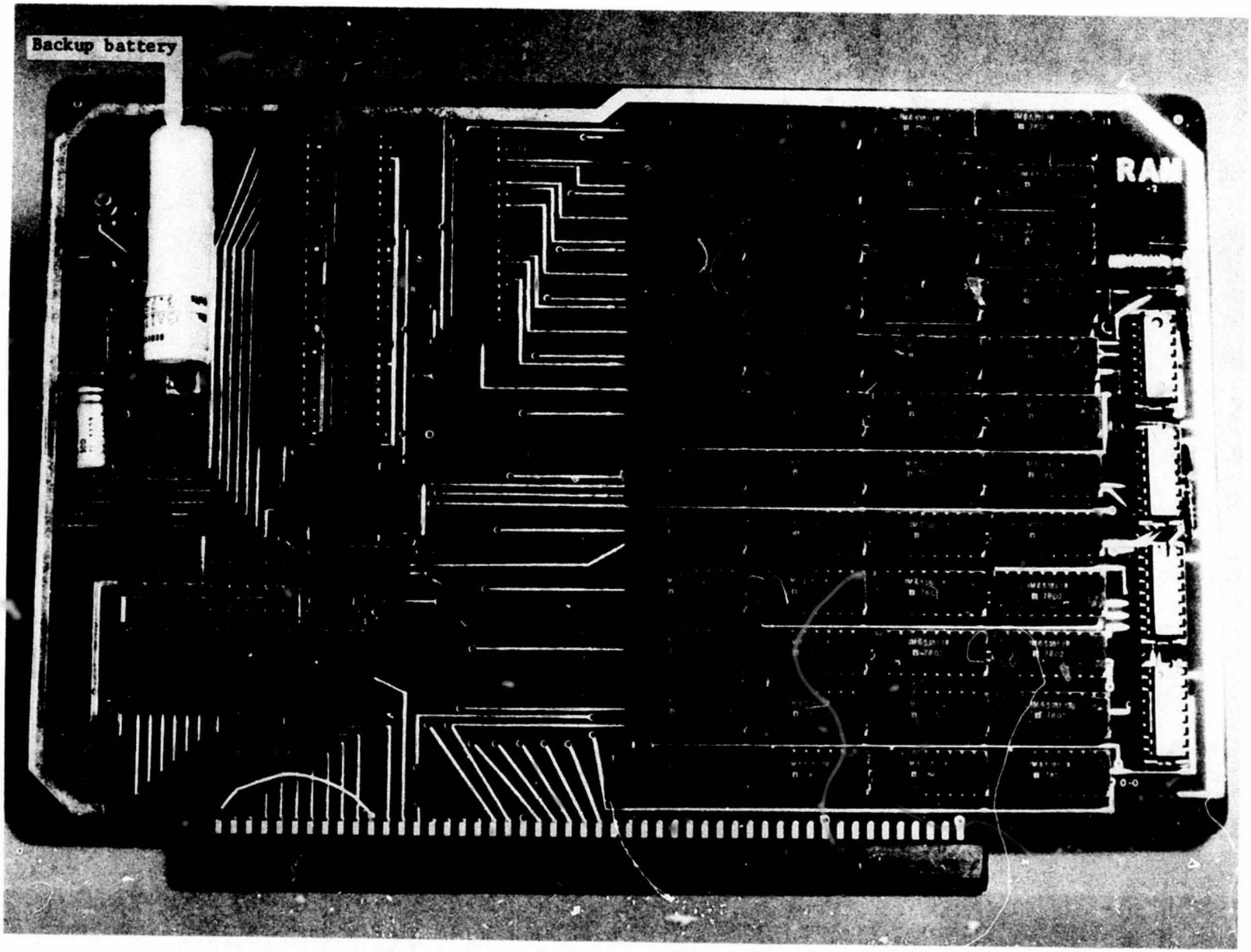
7. Bottom view of card cage showing a closeup of the backplane.



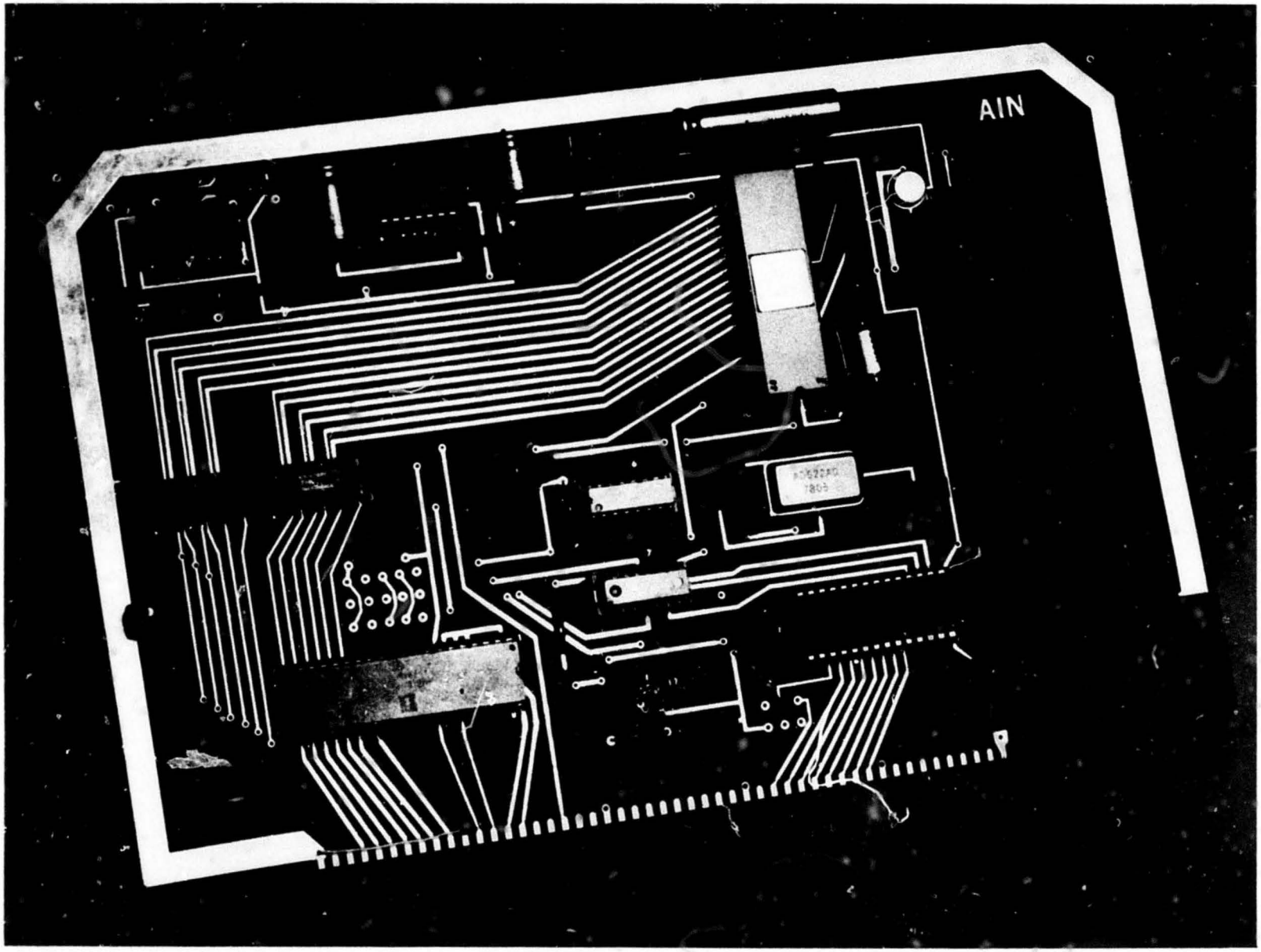
8. CPU Card



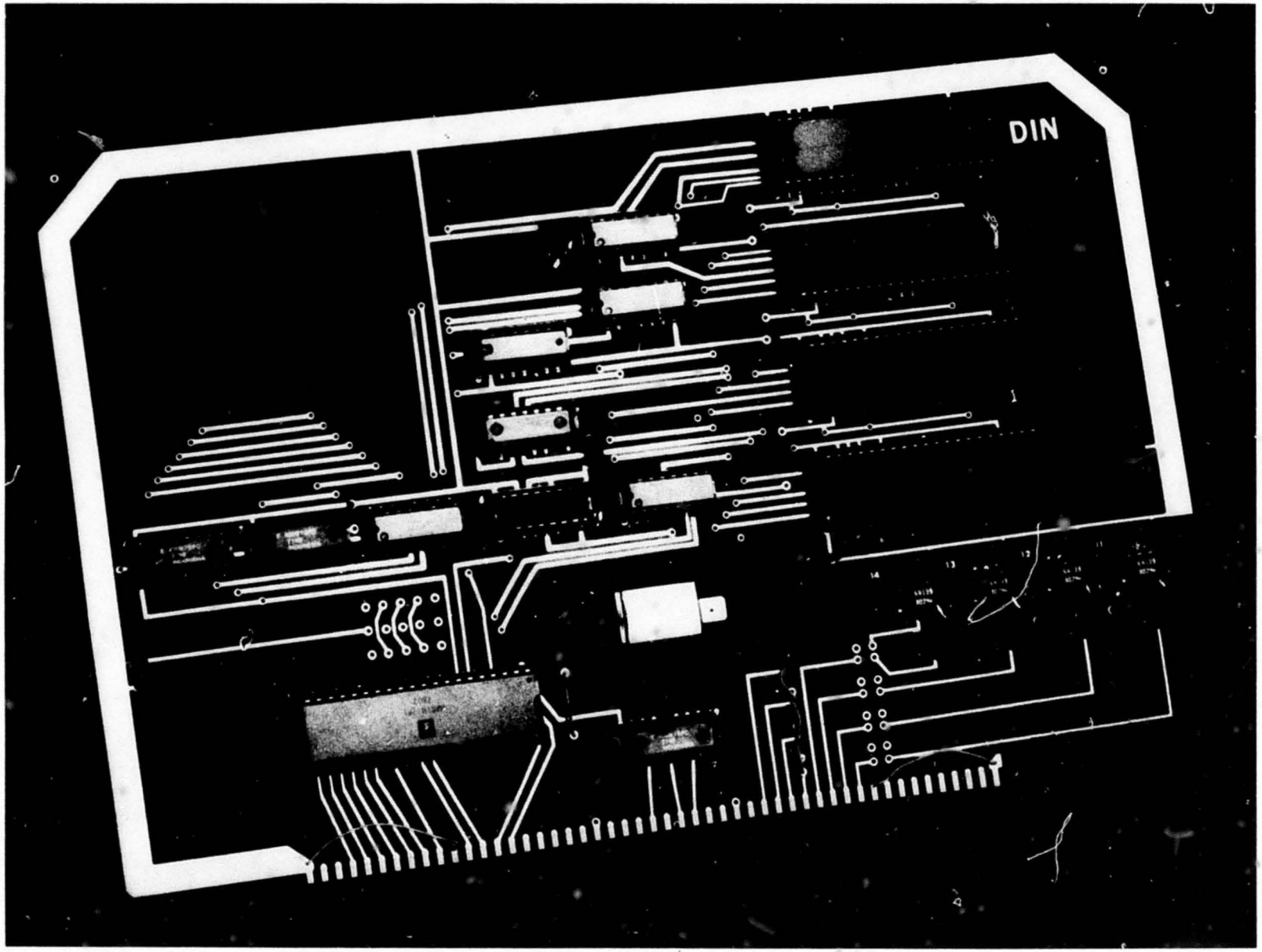
9. Status card



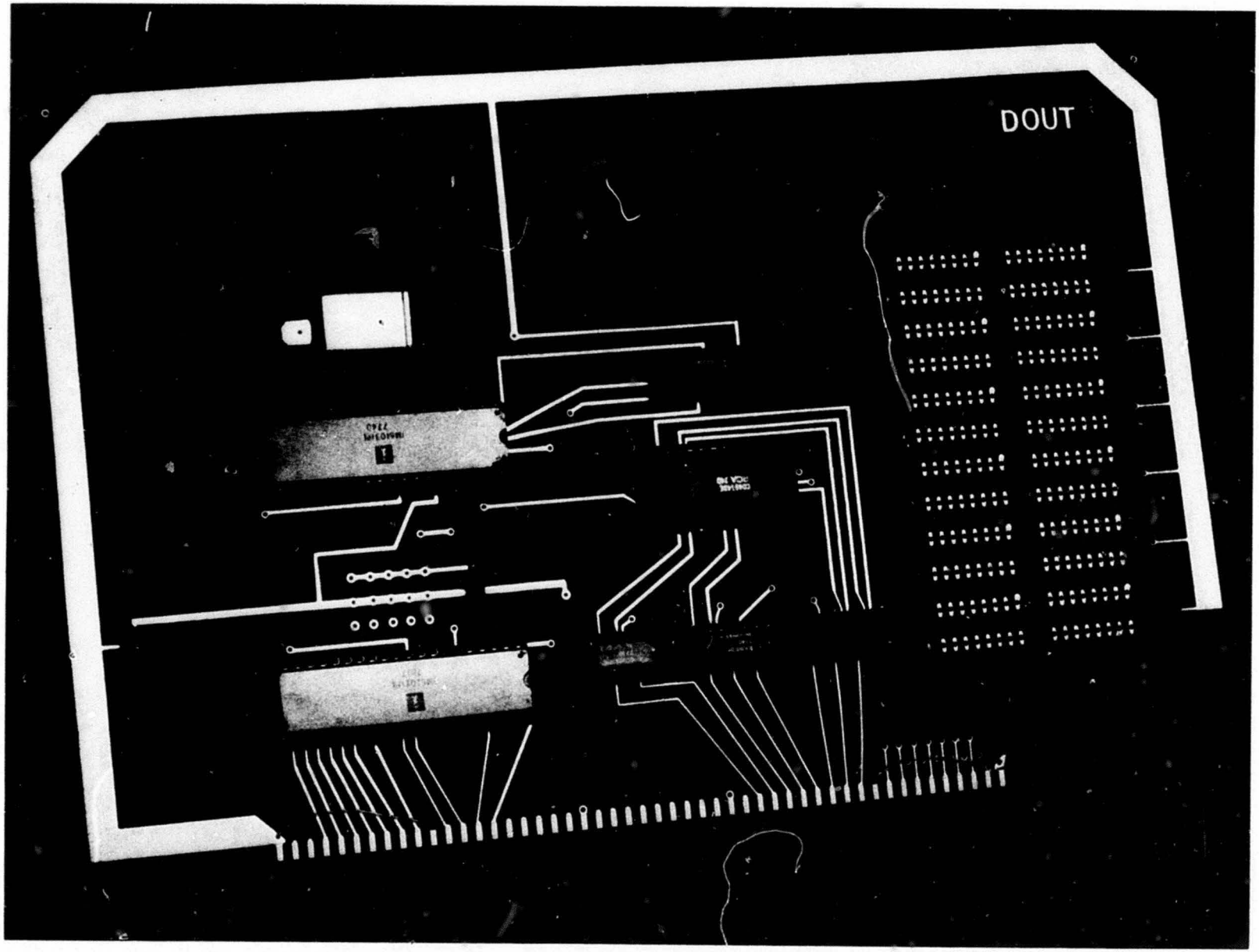
10. Data RAM card



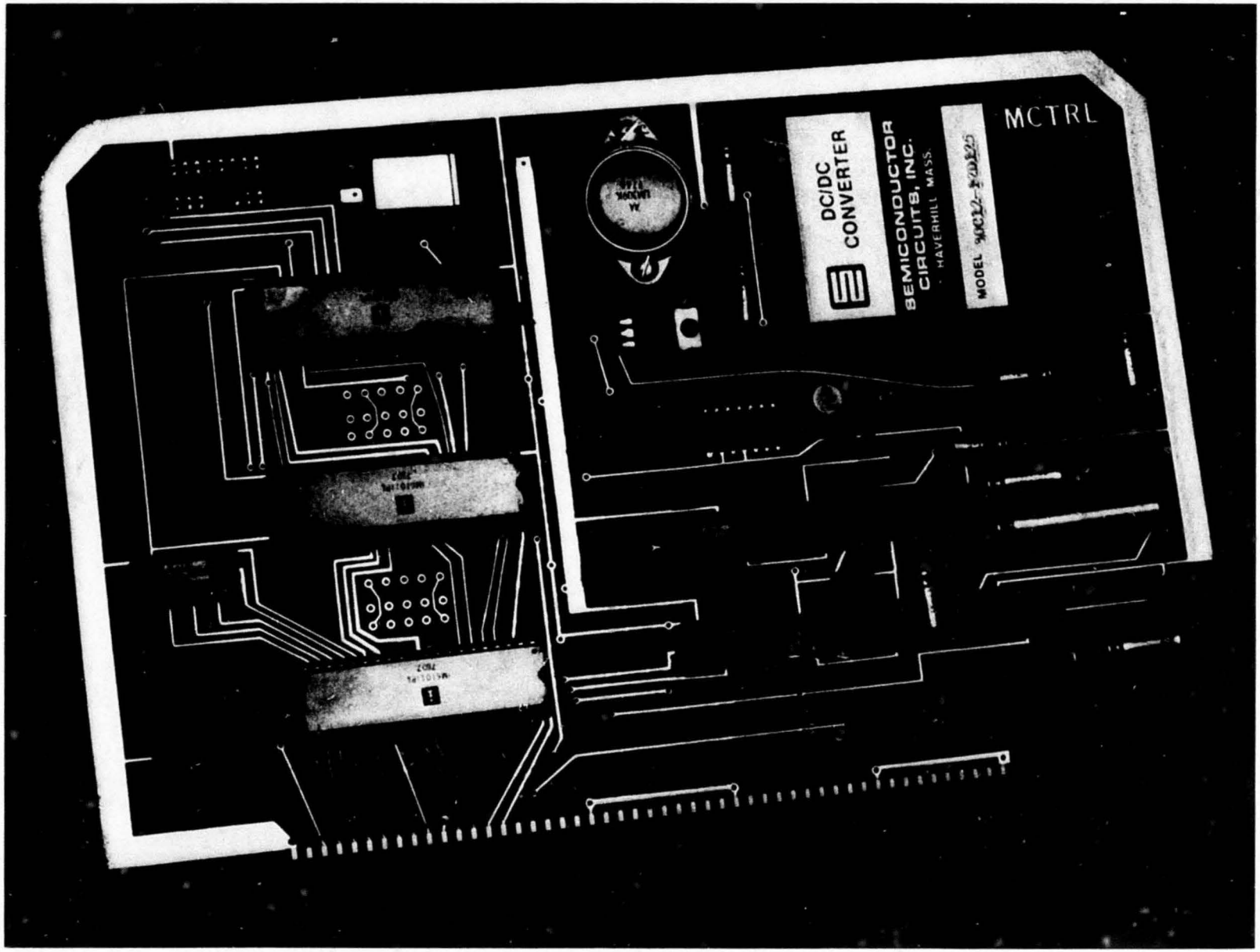
11. Analog input card



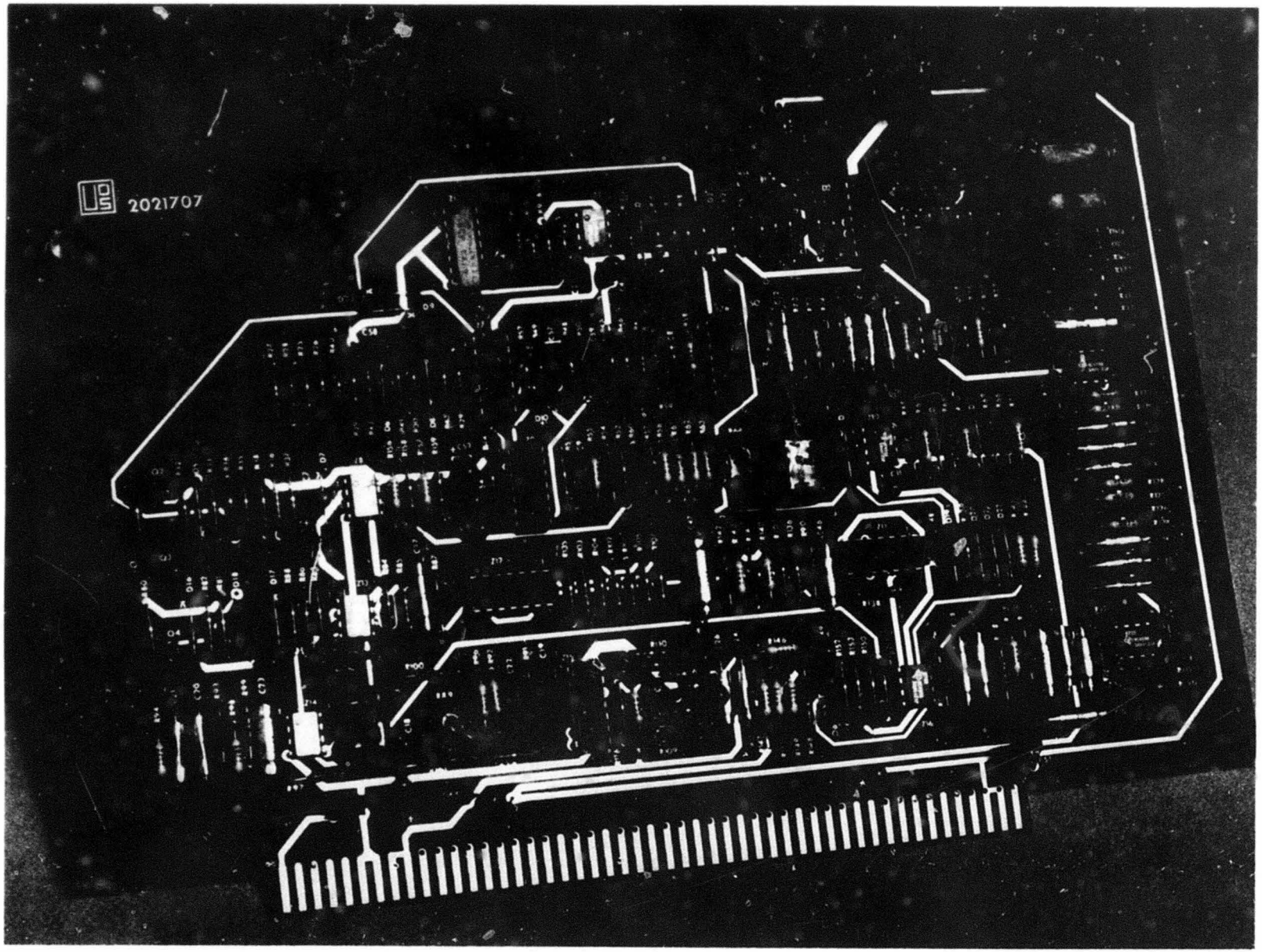
12. Digital input card



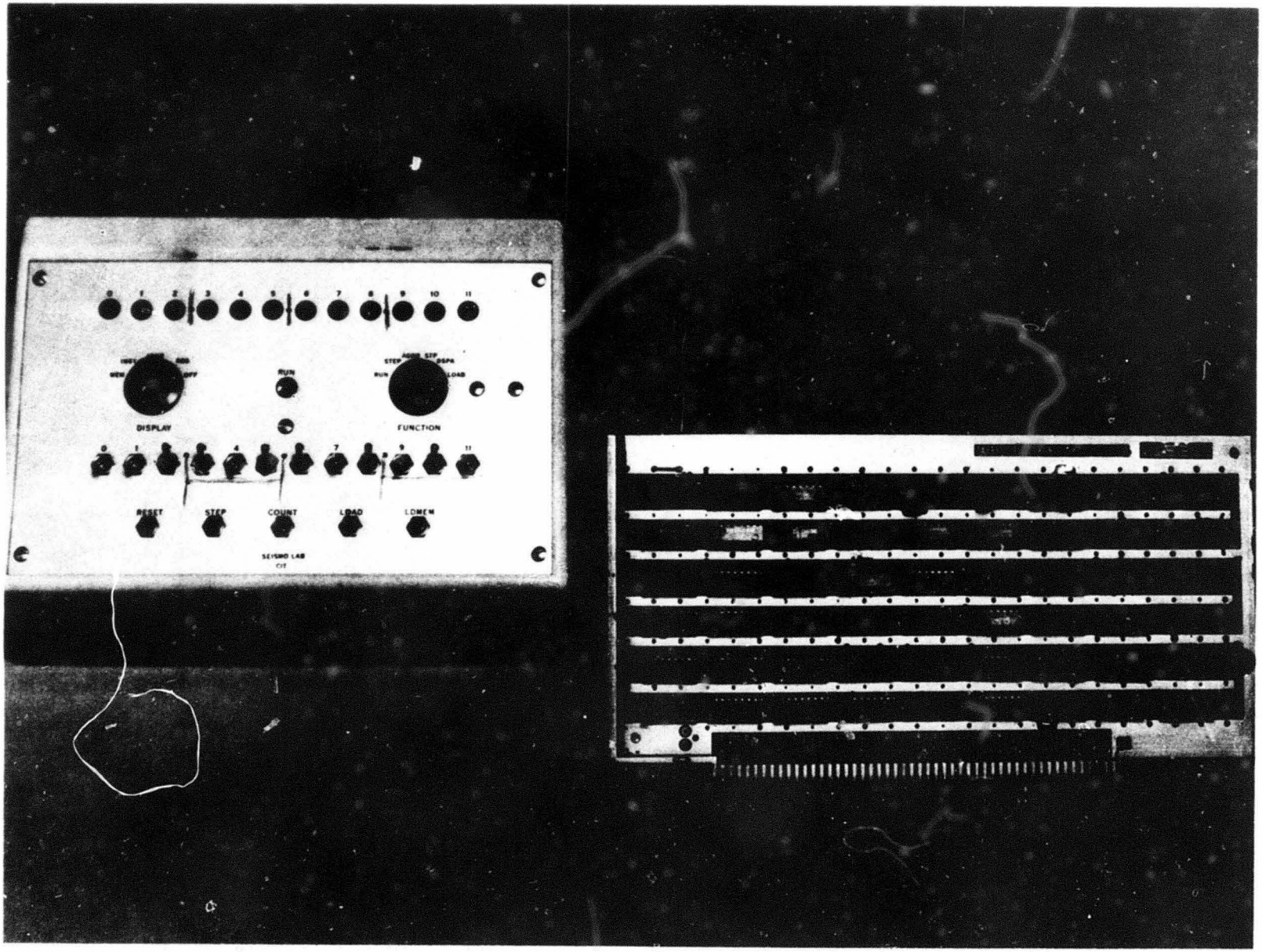
13. Digital output card



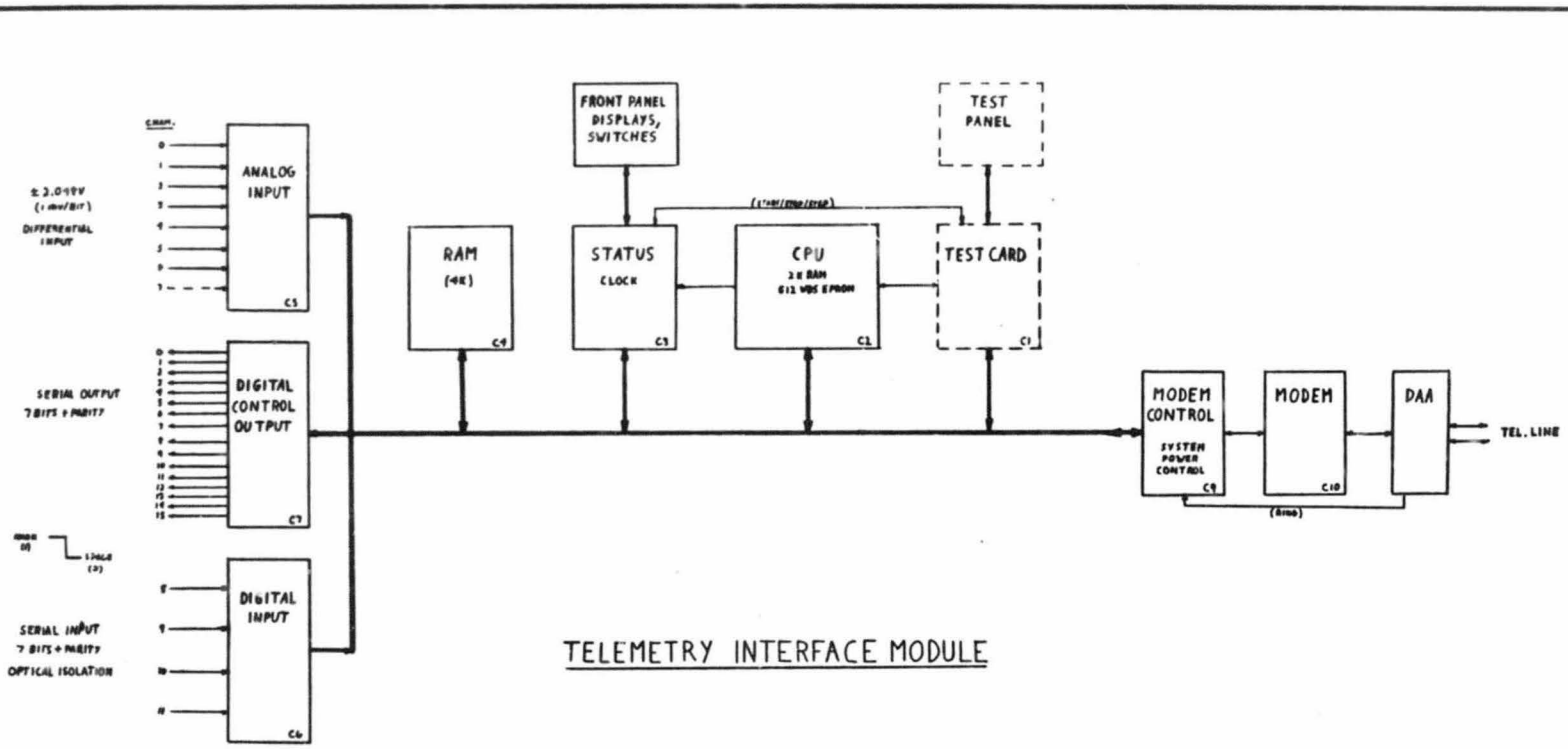
14. Modem controller card



15. Modem card


DISPLAY
RUN
FUNCTION
RESET
STEP
COUNT
LOAD
LOMEM
SEISMO LAB

16. Test box

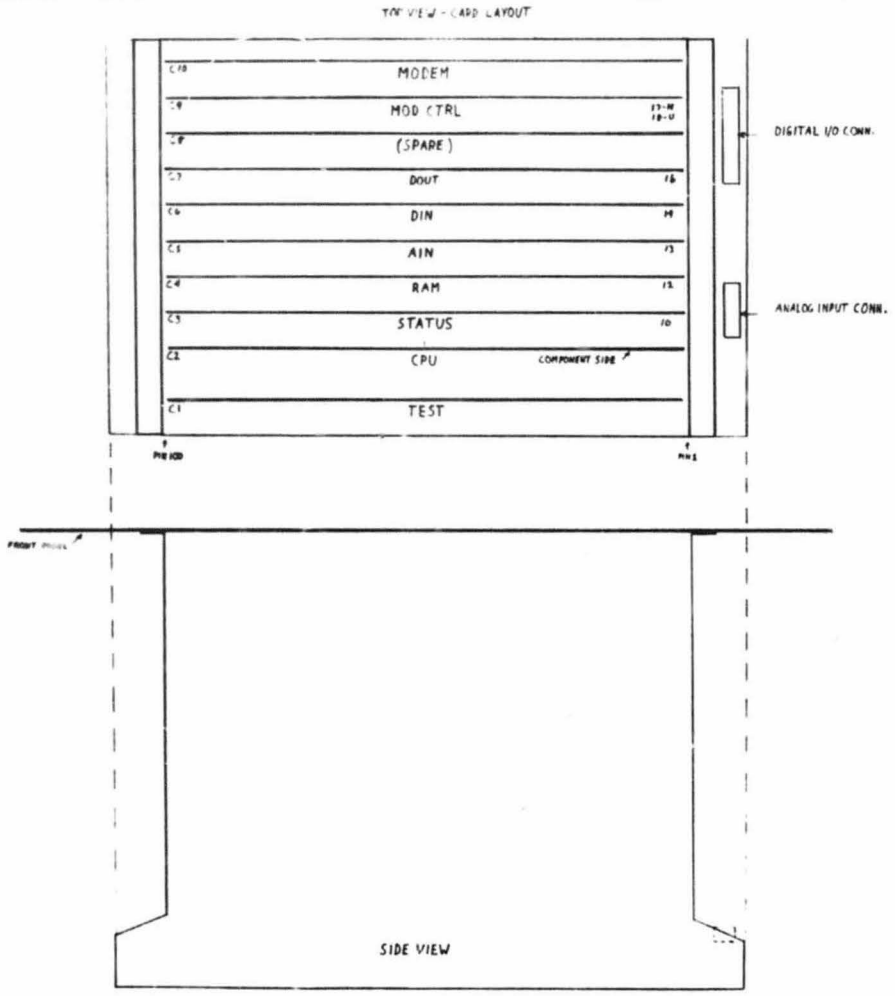


TELEMETRY INTERFACE MODULE

DIGITAL ENGINEERING CO.
WHITTIER, CALIFORNIA

TIM
SYSTEM BLOCK DIAGRAM

DRAWN BY
TRACED BY
APPROVED BY
DATE 7/1/78 DWG. NO. 1



DIGITAL ENGINEERING CO.

WHITTIER, CALIFORNIA

RACK LAYOUT

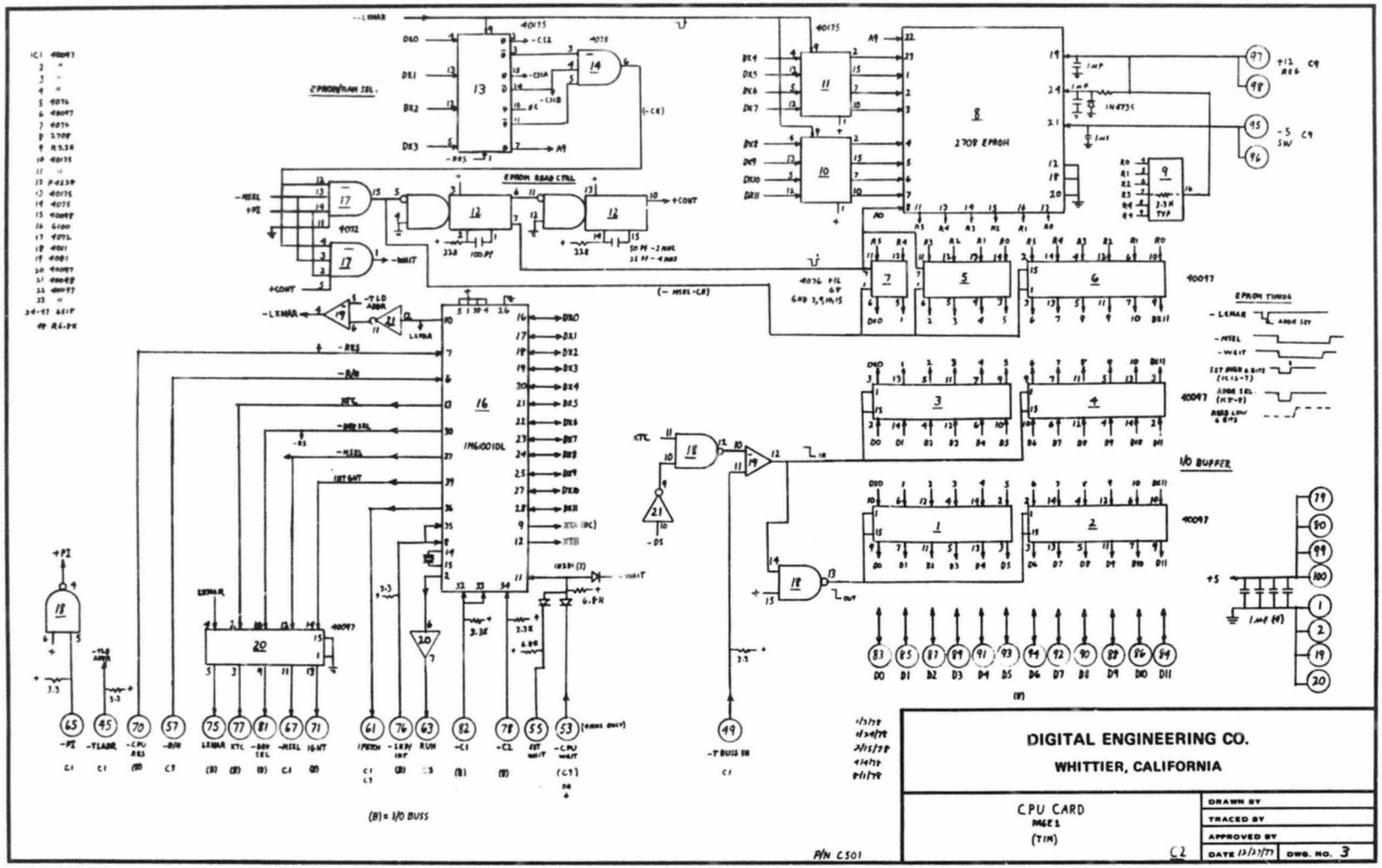
CALTECH TIM

DRAWN BY

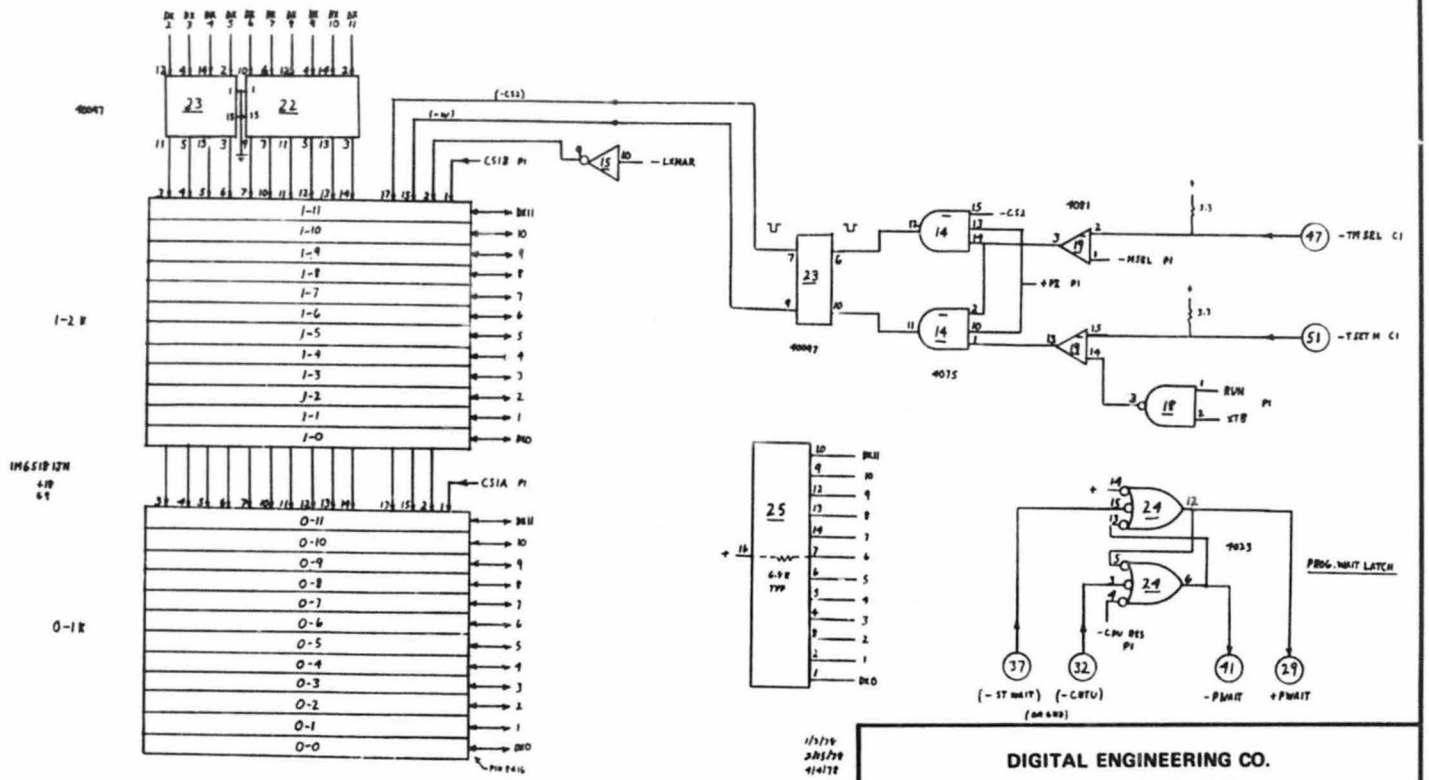
TRACED BY

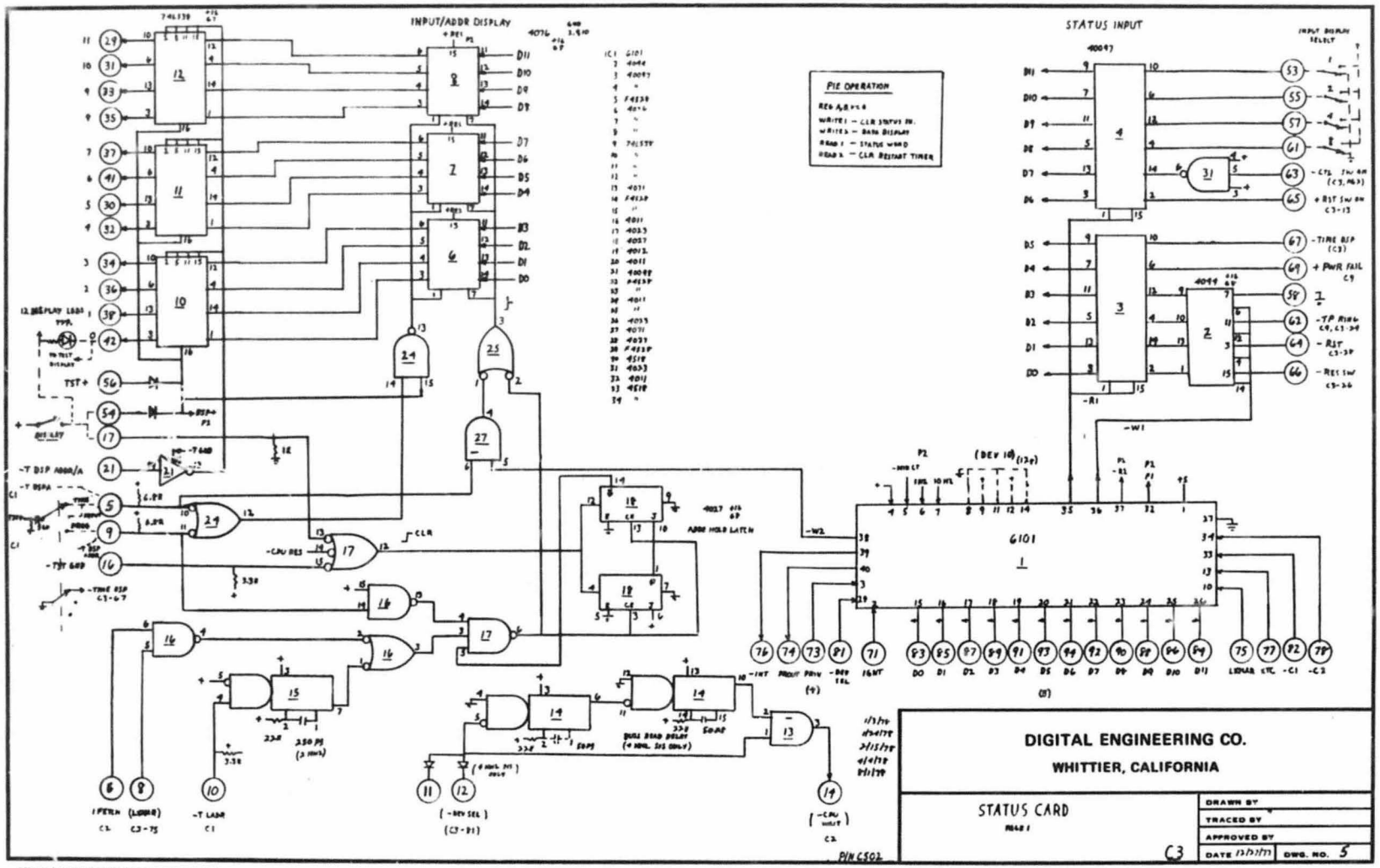
APPROVED BY

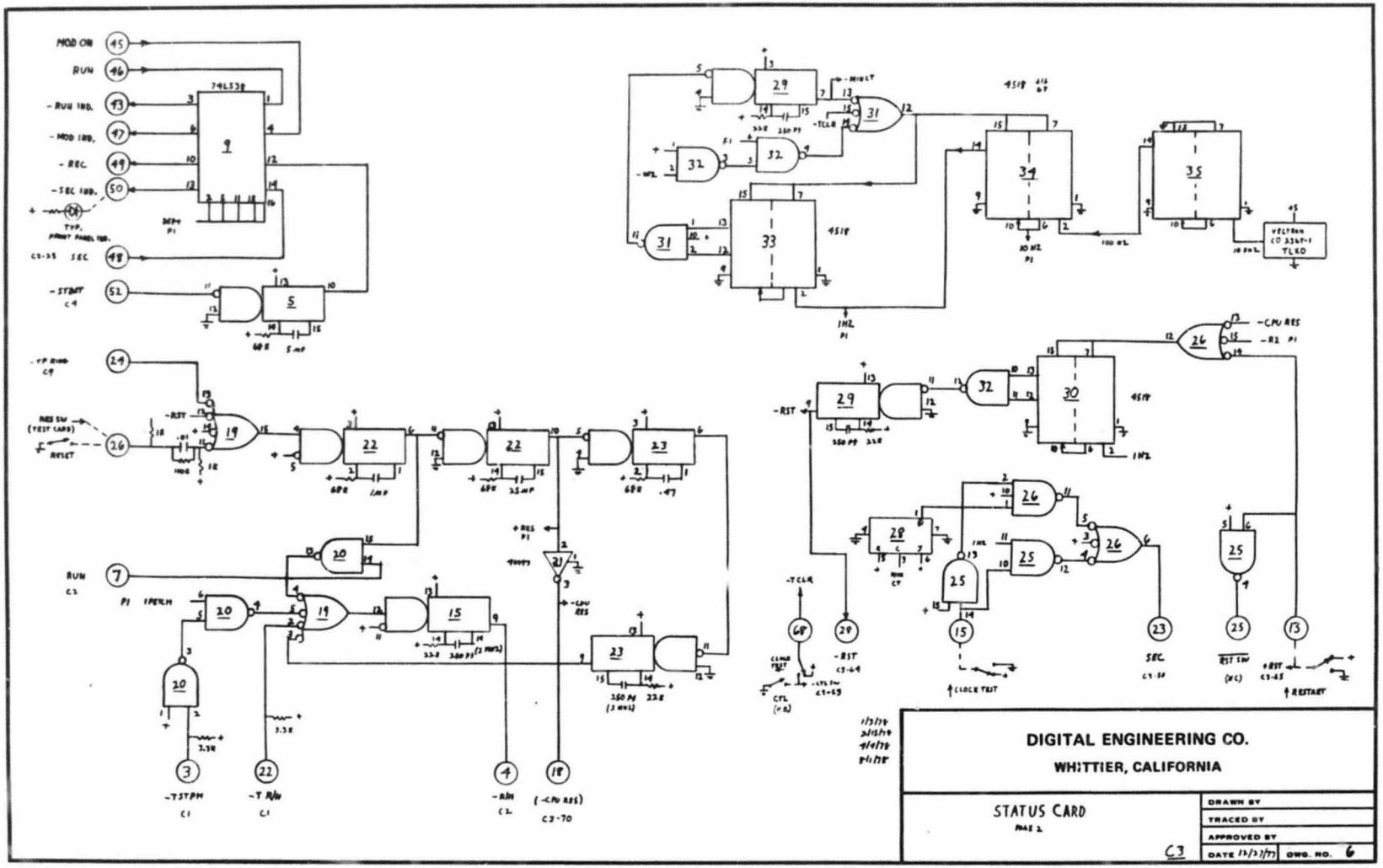
DATE 1/3/74 DWG. NO. 2



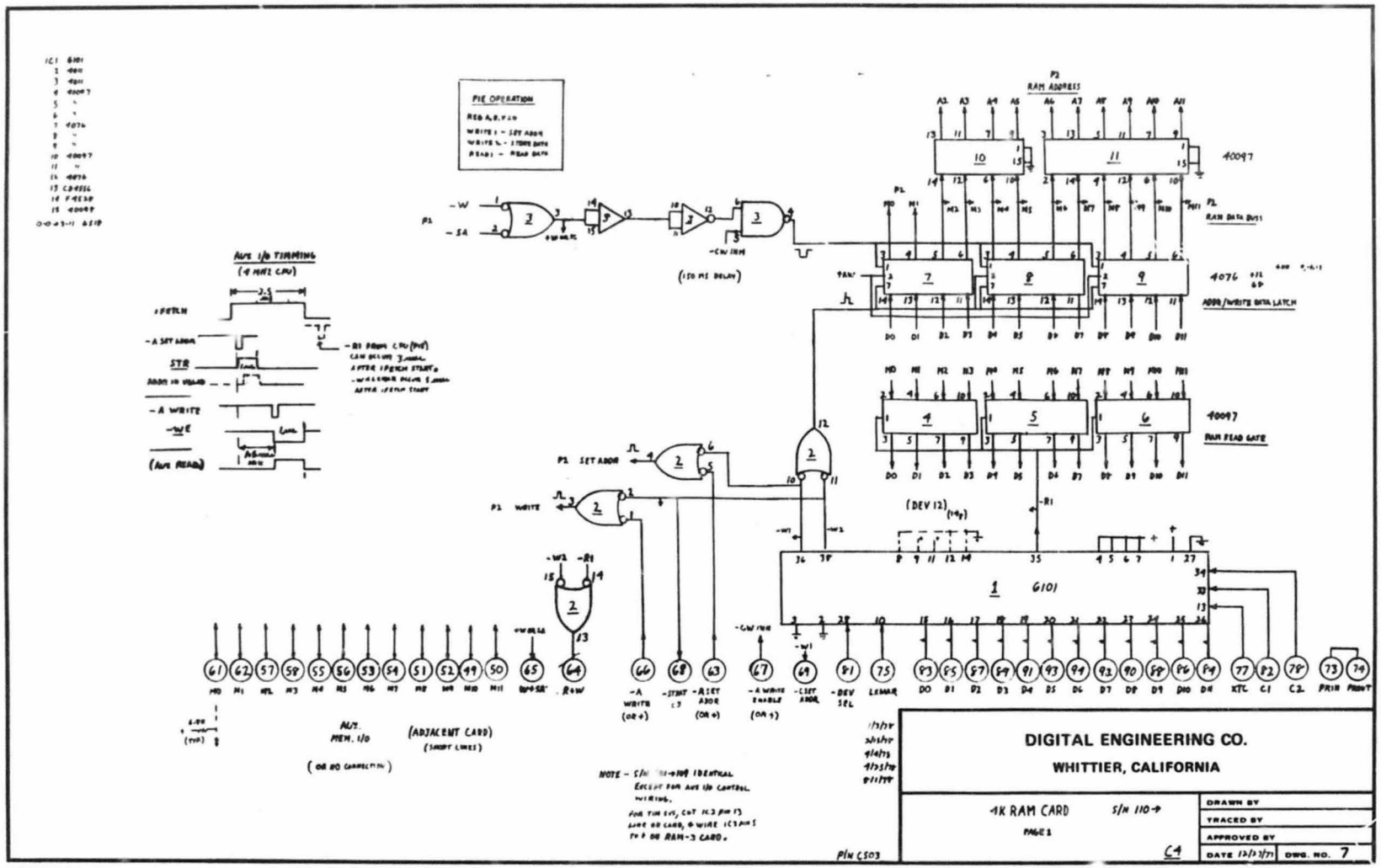
8/77 DIETERICH-POST CLEARPRINT 1000H

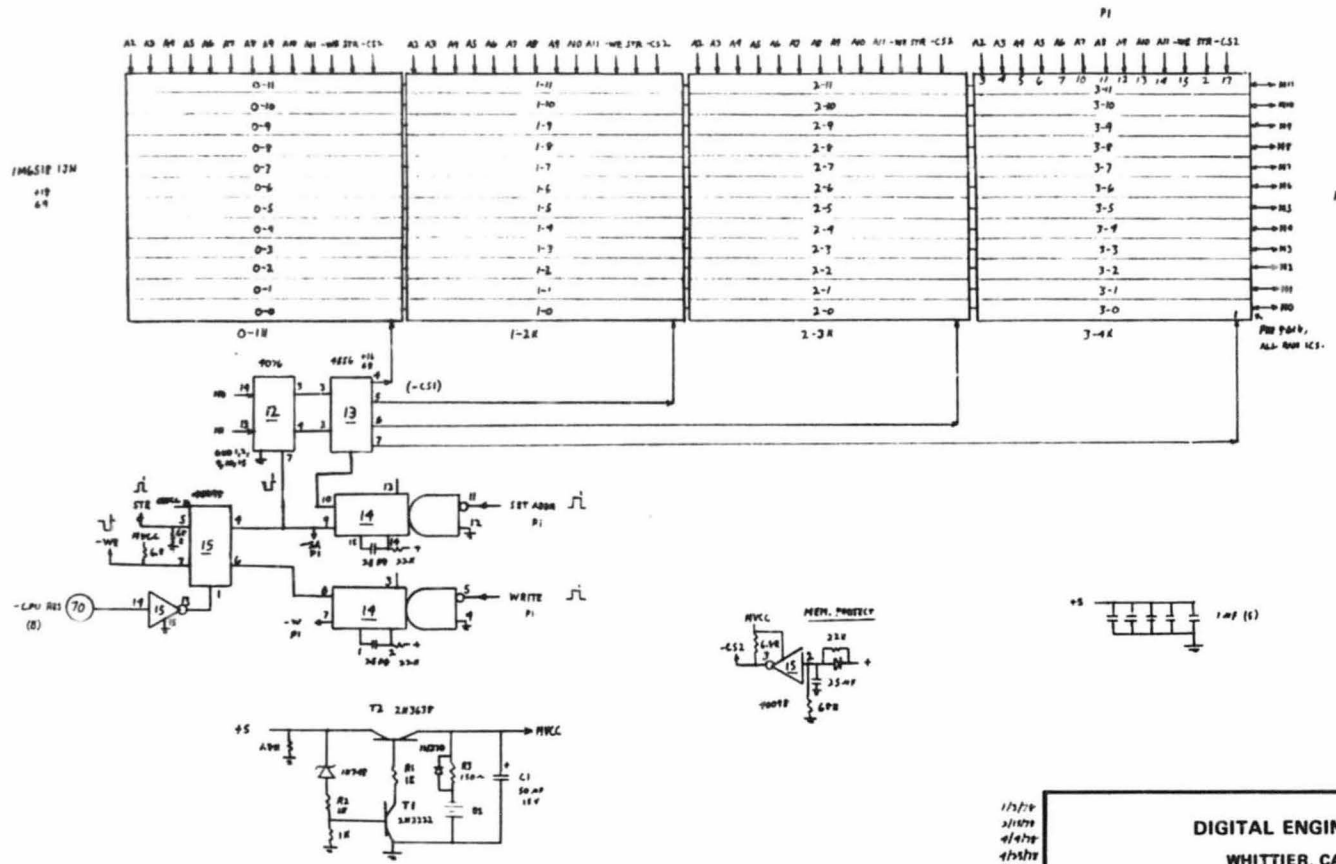






8/77 DIETERICH-POST CLEARPRINT 1000H





DIGITAL ENGINEERING CO.

WHITTIER, CALIFORNIA

4K RAM CARD

PAGE 2

DRAWN BY

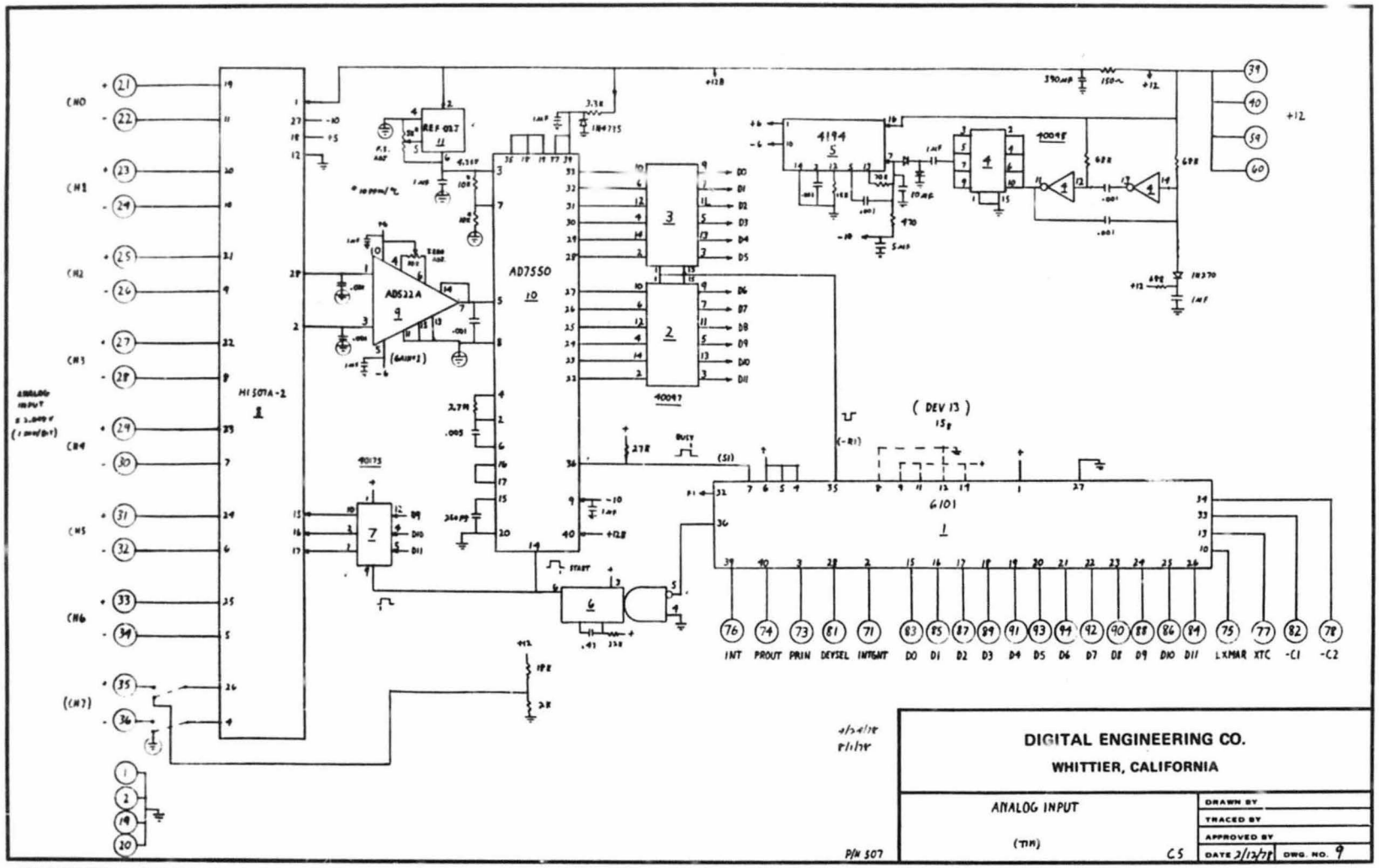
TRACED BY

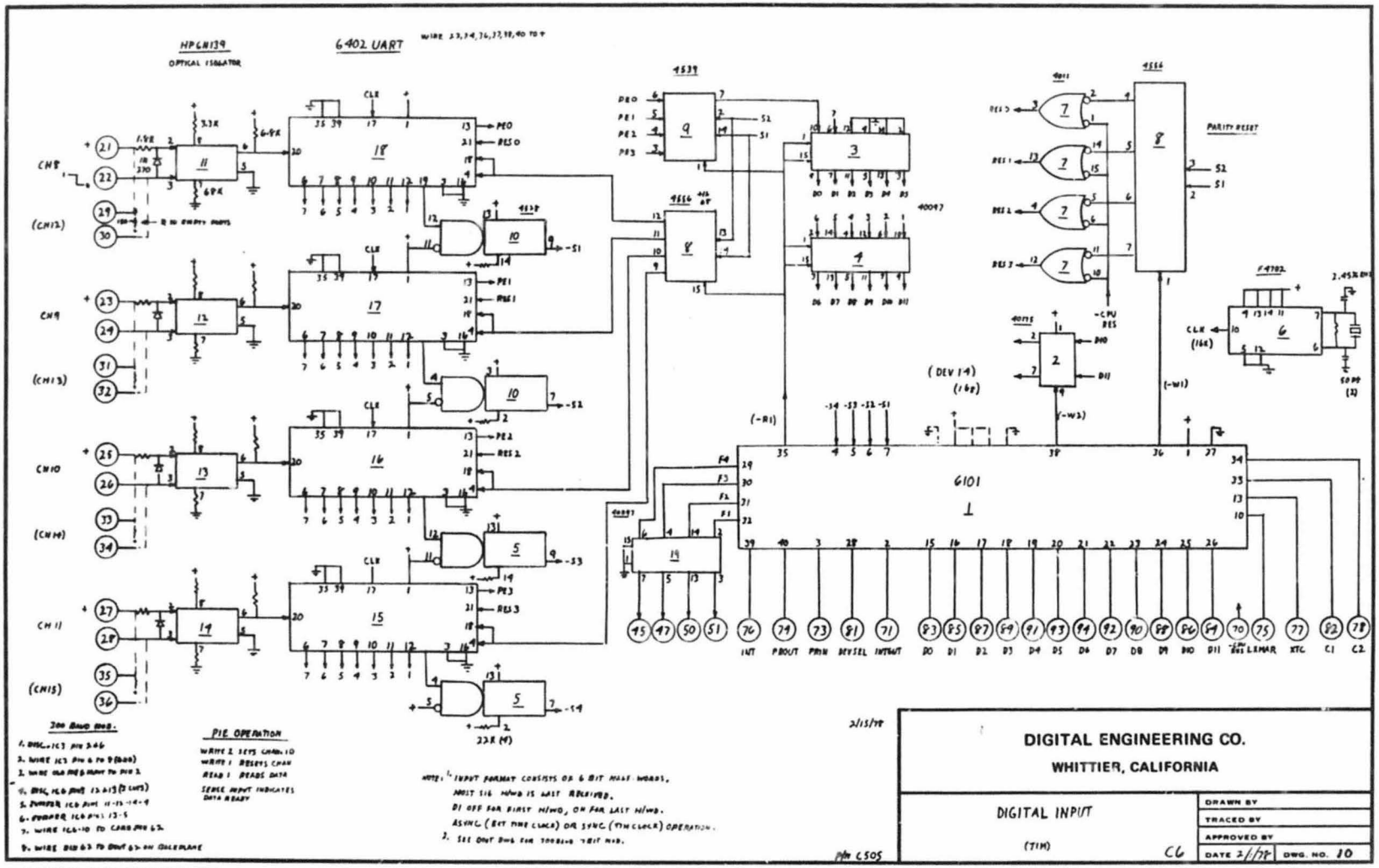
APPROVED BY

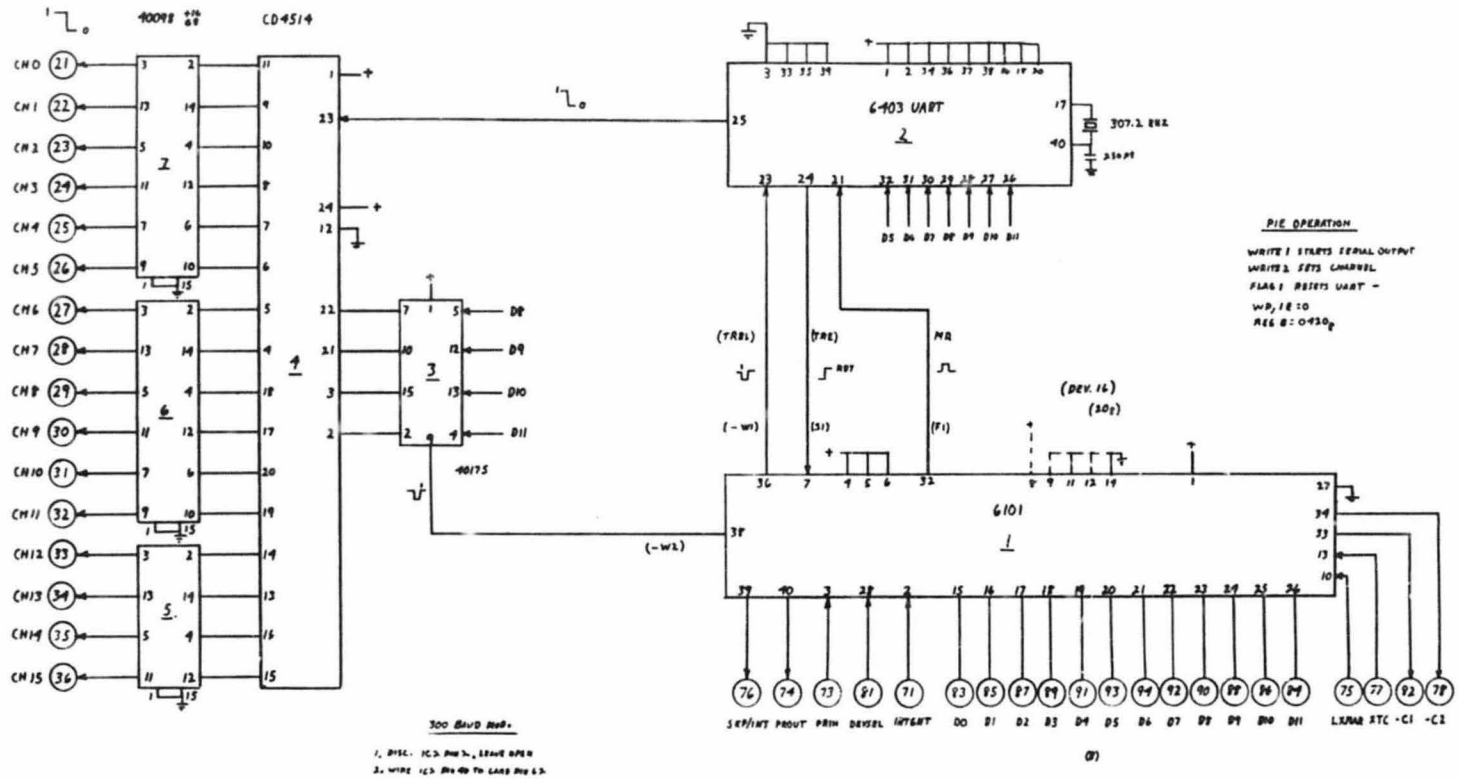
DATE 12/27/77 DWG. NO. 8

C4

8/77 DIETERICH-POST CLEARPRINT 1000H


HI507A-2
AD7550
AD522A
4194
4099
6101
DEV 13
INT
PROUT
PRIN
DEVSEL
INTGNT
LXMAR
XTC
-C1
-C2
DIGITAL ENGINEERING CO.
WHITTIER, CALIFORNIA
ANALOG INPUT
(TIM)
DRAWN BY
TRACED BY
APPROVED BY
DATE 2/12/78
DWG. NO. 9
P/N 507
C5

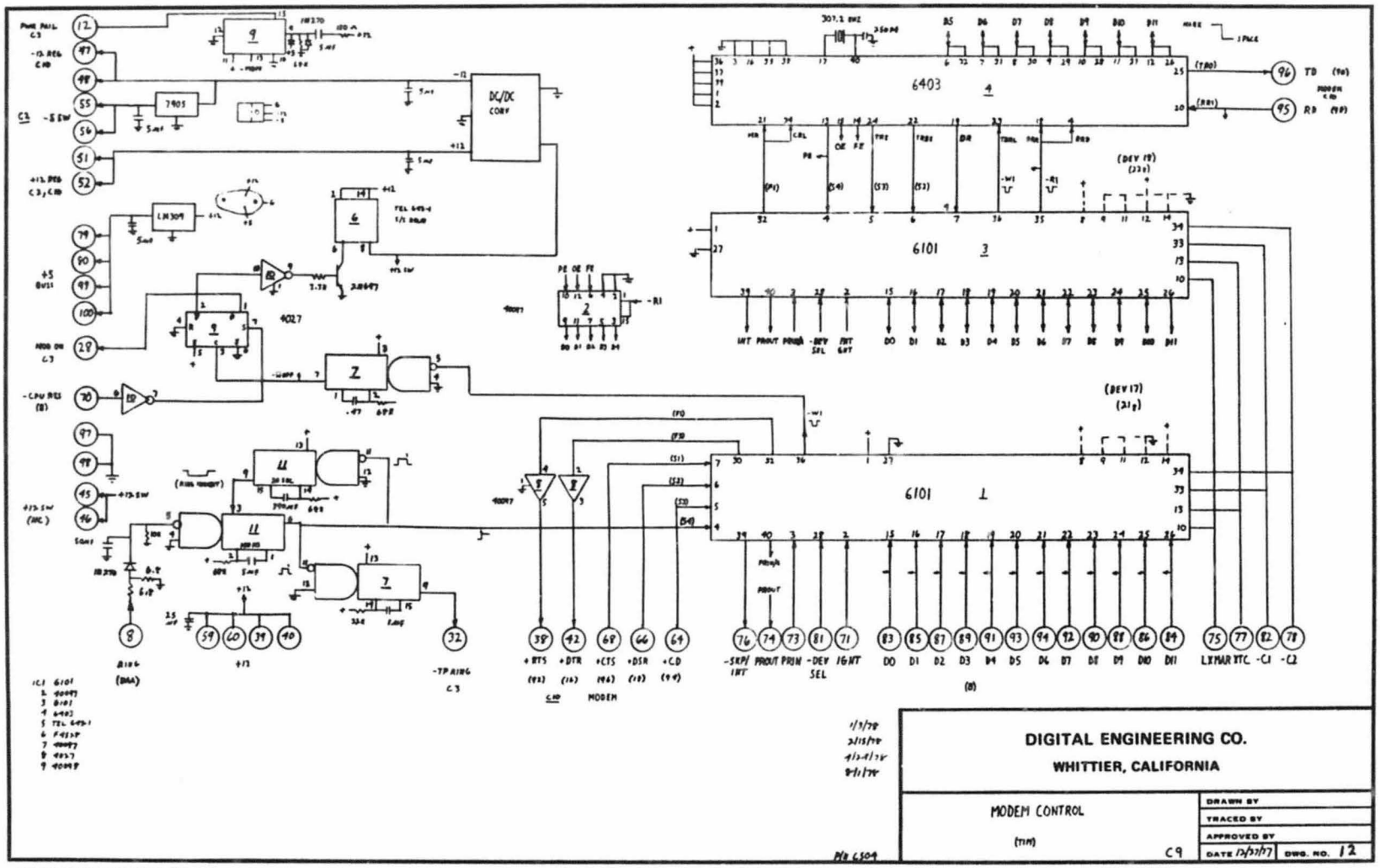

HP6N139
OPTICAL ISOLATOR
6402 UART
CH8
CH9
CH10
CH11
(CH12)
(CH13)
(CH14)
(CH15)
6101
DIGITAL ENGINEERING CO.
WHITTIER, CALIFORNIA
DIGITAL INPUT
(TIM)
C6
DRAWN BY
TRACKED BY
APPROVED BY
DATE 2/15/79
DWG. NO. 10
2/15/79
PIE OPERATION
300 BAUD BUS.
8/77 DIETERICH-POST CLEARPRINT 1000H



DIGITAL ENGINEERING CO.	
WHITTIER, CALIFORNIA	
DIGITAL OUTPUT	DRAWN BY
(TIM)	TRACED BY
CS	APPROVED BY
	DATE 1/20/78 DWG. NO. 11

AN C506

8/77 DIETERICH-POST CLEARPRINT 1000H



8/77 DIETERICH-POST CLEARPRINT 1000H

1	GND	2	GND
19	GND	20	GND
39	+12	40	+12
59	+12	60	+12
69	—	70	-CPU RESET
71	IGNT	72	—
73	(PRIN) —	74	(PROUT) —
75	LXMAR	76	-SKP/INT
77	XTC	78	-C2
79	+5	80	+5
81	-DEVSEL	82	-C1
83	DO	84	DII
85	DI	86	DIO
87	D2	88	D9
89	D3	90	D8
91	D4	92	D7
93	D5	94	D6
95	—	96	—
97	—	98	—
99	+5	100	+5

(12V BATTERY)

← OUTPUT LINE FROM STATUS CARD FOR SYSTEM RESET

PDP-8/I PIE COMPATIBLE BUSS

— = NOT BUSSED

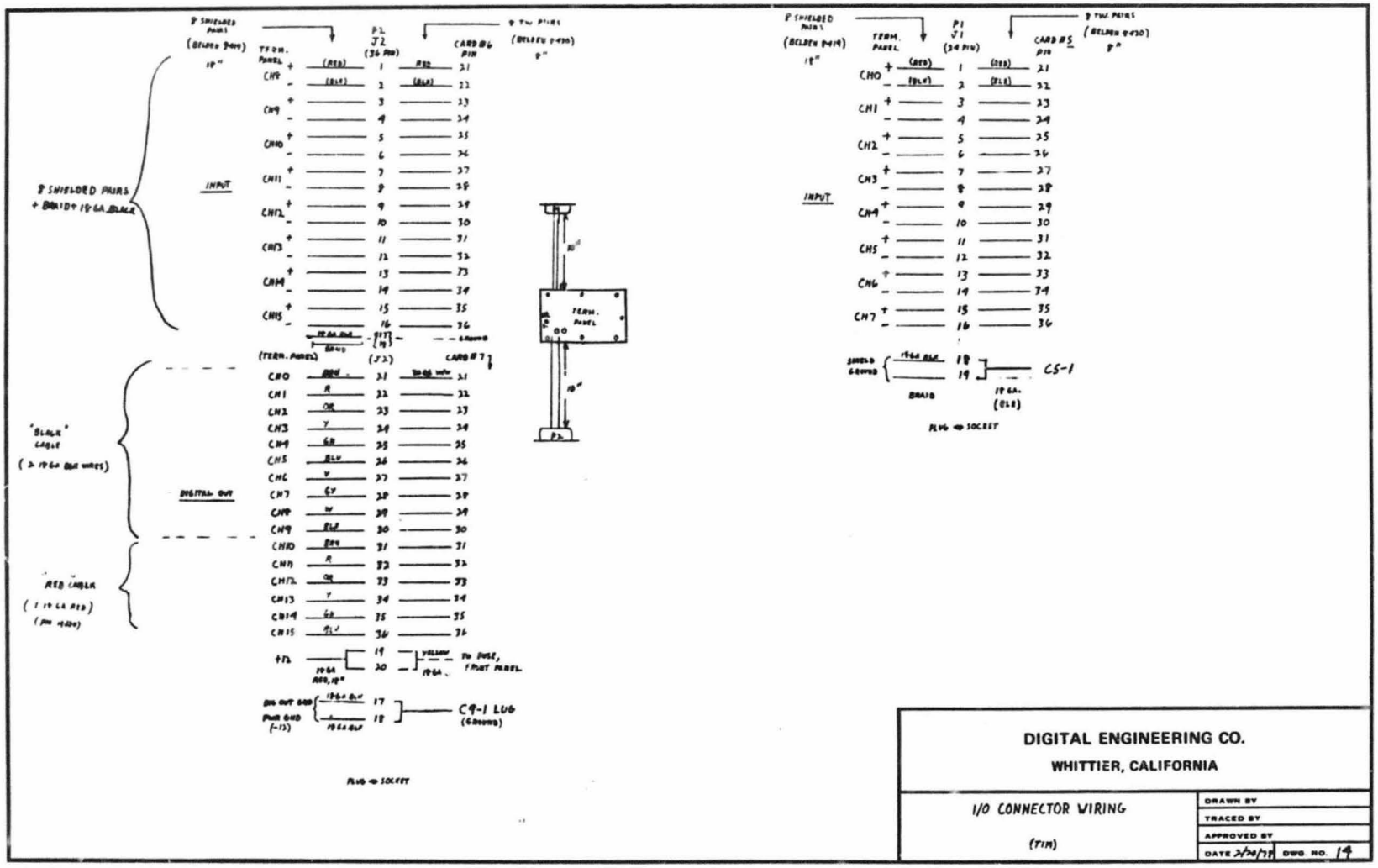
2/15/78
4/4/78

DIGITAL ENGINEERING CO.
WHITTIER, CALIFORNIA

BUSS ASSIGNMENT

(TM)

DRAWN BY	
TRACED BY	
APPROVED BY	
DATE 4/3/78	DWG. NO. 13



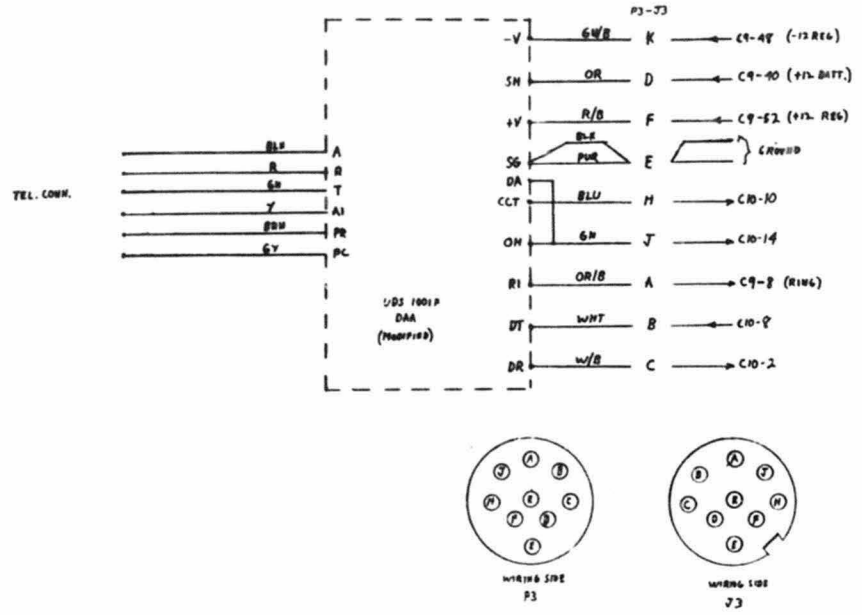
DIGITAL ENGINEERING CO.

WHITTIER, CALIFORNIA

I/O CONNECTOR WIRING

(TIM)

DRAWN BY	
TRACED BY	
APPROVED BY	
DATE 2/20/79	DWG. NO. 19



DIGITAL ENGINEERING CO.		
WHITTIER, CALIFORNIA		
DAA WIRING	DRAWN BY	
	TRACED BY	
	APPROVED BY	
(TIM)	DATE	DWG. NO. 45

```
(0001)  C               ****** X-TIM.TALK ******
(0002)
(0003)  C*****************************************************************************
(0004)  C                                                                            *
(0005)  C     This version of "X-TIM.TALK" is designed to run on a PRIME computing    *
(0006)  C     system. It makes use of a vadic auto-dialer modem. Since full data-set  *
(0007)  C     control of the modem is needed and not supplied or supported a special  *
(0008)  C     data-set controller with appropiate implementing subroutines was designed *
(0009)  C     and used. It uses an additional communications line to achieve full data- *
(0010)  C     set control.                                                            *
(0011)  C
(0012)  C*****************************************************************************
(0013)
(0014)
(0015)
(0016)
(0017)  C*****************************************************************************
(0018)  C                                                                            *
(0019)  C                       Written by:                                          *
(0020)  C                             Robert F. Nickerson                            *
(0021)  C                             CALTECH Seismological LAboratory               *
(0022)  C                             Pasadena, California                           *
(0023)  C                                                                            *
(0024)  C                             October 31, 1979                               *
(0025)  C                                                                            *
(0026)  C*****************************************************************************
```

```
(0028)  C                          ****** X-TIM.TALK ******
(0029)  C*****************************************************************************
(0030)  C                                                                            *
(0031)  C                                                                            *
(0032)  C                                                                            *
(0033)  C     control word format                                                    *
(0034)  C      bit  1 - (40000) transmit new program                                 *
(0035)  C      bit  2 - (20000) transmit new scan rate list, regardless              *
(0036)  C      bit  3 - (10000) transmit new digital control list, regardless        *
(0037)  C      bit  4 - ( 4000) data dump, program dump, and program                 *
(0038)  C                       load if autoreset occured                            *
(0039)  C      bit  5 - ( 2000) get data RAM contents if over half full              *
(0040)  C      bit  6 - ( 1000) get program RAM contents regardless                  *
(0041)  C      bit  7 - (  400) get program RAM contents only                        *
(0042)  C                       prior to a program load                              *
(0043)  C      bit  8 - (  200) get data RAM contents regardless                     *
(0044)  C      bit  9 - (  100) use short message packets                            *
(0045)  C      bit 10 - (   40) ignor request for call short (not implemented)       *
(0046)  C      bit 13 - (    4) set X-TIM return pointer                             *
(0047)  C      bit 14 - (    2) reset X-TIM clock regs.                              *
(0048)  C      bit 15 - (    1) set data RAM reset flag                              *
(0049)  C                                                                            *
(0050)  C*****************************************************************************
(0051)  C                                                                            *
(0052)  C     explicit FORTRAN channels used                                         *
(0053)  C      5: TIM-station-list                                                   *
(0054)  C      6: program-load-file                                                  *
(0055)  C      7: dump file                                                          *
(0056)  C      8: log-in file                                                        *
(0057)  C      9: report                                                             *
(0058)  C     10: phantom file unit                                                  *
(0059)  C     11: data file dump                                                     *
(0060)  C     12: clock correction                                                   *
(0061)  C                                                                            *
(0062)  C*****************************************************************************
(0063)  C                                                                            *
(0064)  C     use of flag switches (-Fn ; n=1,8)                                     *
(0065)  C      (1) : manual dial                                                     *
(0066)  C      (2) : select start time                                               *
(0067)  C      (4) : print contents of transmitter array.                            *
(0068)  C      (5) : print incoming characters in octal.                             *
(0069)  C      (6) : print checksum values of incoming mess. packet.                 *
(0070)  C      (7) : print dialer characters                                         *
(0071)  C      (8) : print characters in octal that are loading into the smic buffer *
(0072)  C                                                                            *
(0073)  C*****************************************************************************
(0074)  C                                                                            *
(0075)  C     format of station list entries                                         *
(0076)  C       idnnnnntel.#                                                         *
(0077)  C     where:                                                                 *
(0078)  C       id = two character station identification                             *
(0079)  C       nnnnn = control word                                                 *
(0080)  C       tel.# = telephone number                                             *
(0081)  C          e.g. KR260037956811                                               *
(0082)  C                                                                            *
(0083)  C*****************************************************************************
```

```
(0085)
(0086)
(0087)
(0088)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(0088)  C
(0088)  C   Declare all common variables
(0088)  C
(0088)
(0088)
(0088)       PARAMETER (KSIZE = 25) /* number of TIM stations, determines size of relev
(0088)
(0088)       INTEGER*2 TEMP,TEMPH,TEMPL, /* temporary variables
(0088)      $ TIME(4),      /* eight character array for fetching date & time info.
(0088)      $ CKSUM,        /* accumulates checksum of message packets
(0088)      $ PKSIZ,        /* size of message packet
(0088)      $ PSFIL(11),    /* array containing name of TIM program module
(0088)      $ DSFIL(11),    /* array containing name of Digital Control List file
(0088)      $ SSFIL(11),    /* array containing name of Scan Rate List file
(0088)      $ PDFIL(6),     /* array containing name of TIM program dump file
(0088)      $ DDFIL(6),     /* array containing name of TIM data dump file
(0088)      $ TPN(13),      /* array containing characters of tel. # acquired from stati
(0088)      $ STX,          /* ASCII control character
(0088)      $ SI,
(0088)      $ ETX,
(0088)      $ SOH,
(0088)      $ EOM,
(0088)      $ ACK,
(0088)      $ SYNC,        /* Transmission sync. character
(0088)      $ SDATA,       /* TIM op. code - send data dump
(0088)      $ SPROG,       /* TIM op. code - send program dump
(0088)      $ SSTAT,       /* TIM op. code - send TIM system status
(0088)      $ LPROG,        /* TIM op. code - load program module
(0088)      $ LPRGC,       /* TIM op. code - load program module, continuation
(0088)      $ LSRL,        /* TIM op. code - load scan rate list
(0088)      $ LDCL,        /* TIM op. code - load digital control list
(0088)      $ TERM,        /* TIM op. code - terminate telemetry
(0088)      $ ERCODE,      /* error return argument
(0088)      $ STNID,       /* 2 character TIM station id. code
(0088)      $ WORD(4100), /* array to receive TIM data dump
(0088)      $ CWORD,       /* control word, controls program flow
(0088)      $ XSTAT,       /* receives TIM system status word
(0088)      $ OCTDCD,      /* integer function subroutine
(0088)      $ CTLLIN,      /* AMLC line for MODEM set control
(0088)      $ COMLIN,      /* AMLC line for telephone telemetry
(0088)      $ NTRY         /* number of trys before aborting station
(0088)
(0088)       INTEGER*4 WAITTM, /* wait time (used with SLEEP$ subroutine)
(0088)      $ DATE(4)      /* array to receive system date in ASCII
(0088)
(0088)       REAL*8 CORR  /* correction to system clock
(0088)
(0088)       LOGICAL CTS$C,DSR$C,CD$C, /* logical functions to control MODEM
(0088)      $ ON,OFF       /* logical parameters to control MODEM
(0088)       LOGICAL FLAG(9) /* control program flow with runtime options
(0088)
(0088)       COMMON /IO/IRPRT /* FORTRAN channel for report file
```

```
(0083)        COMMON /LOGIC/ON,OFF
(0084)        COMMON /FILES/PSFIL,DSFIL,SSFIL,PDFIL,DDFIL
(0085)        COMMON /COMM/TPN,STX,SI,ETX,SOH,EOM,SYNC,ACK,NAK
(0086)        COMMON /OP/SDATA,SPROC,SSTAT.LPROC,LPRCC,LSRL,LDCL,TERM
(0087)        COMMON /PARAM/STNID,CWORD,NTRY,XSTAT,XTIME
(0088)        COMMON /SLOG/CORR,
(0089)       $          MINDX,          /* indexing variable
(0090)       $          MSID(KSIZE),    /* receives station id. list from "X-DATA.LOG.DA"
(0091)       $          ADATE(KSIZE,2), /* receives origin date for current TIM data dump
(0092)       $          OTIME(KSIZE,2),  /* receives origin time for current TIM data dum
(0093)       $          DATE,TIME
(0094)        COMMON /CONST/MWT /*MODEM wait time before abort and error return
(0095)        COMMON WORD
(0096)        COMMON /FLAGS/FLAG
(0097)        COMMON /LINES/CTLLIN,COMLIN
(0098)  C end of COMMON declarations
(0099)
(0100)
(0101)
(0102)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0103)        NOLIST
(0104)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0105)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0106)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0107)        NOLIST
(0108)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(0109)  C Insert file for mnemonic APPLIB keys (FTN)
(0110)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(0111)        NOLIST
(0112)
(0113)        INTEGER CHARAY(6),CHARY(5)
(0114)        INTEGER INFO(8),IBUFF(8),ICHAR(1)
(0115)        REAL*8 KDST
(0116)
(0117)
(0118)  C----------do some initializing----------
(0119)
(0120)  C----set polling time
(0121)        POLLTM = 23.50
(0122)  C----set modem wait time
(0123)        MWT = 3
(0124)
(0125)  C----set daylight saving time offset (0 for standard, -1 for daylight).
(0126)        KDST = 0
(0127)
(0128)  C----pick up any flags from command file
(0129)     130 CALL RDTK$$(1,INFO,IBUFF,8,ICODE)
(0130)         IF (INFO(1).EQ.6) GOTO 120                  /* CHECK FOR END OF LINE
(0131)         IF (INFO(1).NE.1) GOTO 130                  /* CHECK FOR A TOKEN
(0132)         IF (AND(INFO(3),:020000).EQ.0) GOTO 135     /* CHECK FOR CORRECT TOKEN TYPE
(0133)         IF (IBUFF(1).EQ.'-F') GOTO 125
(0134)     135 CALL TNOU('BAD FLAG OPTION',15)
(0135)         CALL EXIT
(0136)     125 ICHAR(1) = IBUFF(2)
(0137)         DECODE(1,25,ICHAR,ERR=135)NFLAG
(0138)      25   FORMAT(I1)
```

```
(0119)         IF ((NFLAG.LT.1).OR.(NFLAG.GT.9).OR.(ICODE.NE.0)) GOTO 135
(0120)         FLAG(NFLAG) = .TRUE.
(0121)         GOTO 130
(0122)
(0123)  C----if flag-3 is set, go to sleep till 2330 hours
(0124)     120 IF (.NOT.FLAG(3)) GOTO 140
(0125)         WAITTM = (PGLLTM-TIME$A(TIME))*3600.*1000.
(0126)         CALL SLEEP$(WAITTM)
(0127)
(0128)  C----get time correction
(0129)     140 CALL OPEN$A(A$READ,'CLKCOR',6,8)
(0130)         READ(12)CORR
(0131)         CALL CLOS$A(8)
(0132)  C----zero the time log arrays
(0133)         MINDX = 0
(0134)         DO 160 I=1,KSIZE
(0135)           MSID(I) = 0
(0136)           DO 160 J=1,2
(0137)             ADATE(I,J) = 0
(0138)             OTIME(I,J) = 0.0
(0139)     160     CONTINUE
(0140)
(0141)         IF (.NOT.FLAG(2)) GOTO 150
(0142)
(0143)  C----wait until start time
(0144)         CALL TNOUA('TYPE IN START TIME 'HHMM' : ',27)
(0145)         READ(1,15) HOUR,BMIN
(0146)      15   FORMAT(2F2.0)
(0147)         WAITTM = ((HOUR+BMIN/60.)-TIME$A(TIME))*1000.*3600.
(0148)         IF (WAITTM.LE.0) GOTO 150
(0149)         CALL SLEEP$(WAITTM)
(0150)
(0151)
(0152)
(0153)
(0154)
(0155)
(0156)
(0157)  C----open report file to append
(0158)     150 IF (OPEN$A(A$RDWR+A$SAMF,'REPORT.TX',9,5)) GOTO 155
(0159)         CALL TNOU('COULD NOT OPEN REPORT FILE',26)
(0160)         CALL CLSOUT
(0161)     155 IF (GEND$A(5)) GOTO 165
(0162)         CALL TNOU('COULD NOT FIND E-O-F OF REPORT FILE',34)
(0163)         CALL CLSOUT
(0164)
(0165)  C----force new page on report file
(0166)     165 NPG = 4
(0167)
(0168)  C----open station log-in file
(0169)         IF (OPEN$A(A$READ,'X-DATA.LOG.DA',13,4)) GOTO 170
(0170)         CALL TNOU('COULD NOT OPEN TIME LOG FILE',28)
(0171)         CALL CLSOUT
(0172)
(0173)  C----fetch it
(0174)     170 READ(8,END=180,ERR=180) (MSID(I),ADATE(I,1),OTIME(I,1),I=1,KSIZE)
```

```
(0175)     180 CALL CLOS@A(4)
(0176)
(0177)   C----be sure old time references are available if needed
(0178)   C---- in the future
(0179)         DO 190 I=1,KSIZE
(0180)           ADATE(I,2) = ADATE(I,1)
(0181)           OTIME(I,2) = OTIME(I,1)
(0182)     190   CONTINUE
(0183)
(0184)
(0185)   C----set for number of trys at message exchange with
(0186)   C----marginal linkup
(0187)         NTRY = 5
(0188)
(0189)   C----XLOAD and XDUMP argument definitions
(0190)         KPROG = 1      /* LOAD PROGRAM
(0191)         KDCL  = 2      /* LOAD DIGITAL-CONTROL-LIST
(0192)         KSRL  = 3      /* LOAD SCAN-RATE-LIST
(0193)         KDATA = 2
(0194)
(0195)         CALL OPEN@A(A@READ,'X-TIM.LIST.TX',13,1)
(0196)
(0197)   C-------------------------------------------------------------
(0198)
(0199)
(0200)
(0201)
(0202)
(0203)
(0204)   C----------dial the current station and get its status----------
(0205)
(0206)   C----read in station id, control word, and tp number
(0207)     200 READ(5,10,END=900,ERR=800) STNID,CHARY,(TPN(I),I=1,13)
(0208)      10   FORMAT(A2,5A1,13A1)
(0209)   C----check if comment or inactive station
(0210)         IF(STNID.NE.'* ') GOTO 205
(0211)         WRITE(1,75)(CHARY(I),I=1,2)
(0212)         WRITE(IRPRT,75)(CHARY(I),I=1,2)
(0213)      75   FORMAT(//1X,2A1,' NOT ACTIVE'//)
(0214)         GOTO 200
(0215)     205 CWORD = OCTDCD(CHARY)
(0216)
(0217)   C----check for end of list
(0218)         IF ((STNID.EQ.0).OR.(STNID.EQ.'  ')) GOTO 900
(0219)
(0220)   C----forms report if necessary
(0221)         NPG = NPG+1
(0222)         IF (NPG.LE.3) GOTO 210
(0223)         NPG = 1
(0224)         CALL DATE@A(DATE)
(0225)         CALL TIME@A(TIME)
(0226)         WRITE(IRPRT,80)DATE,TIME
(0227)      80   FORMAT(1H1//1X,'CROSS SYSTEM REPORT FOR ',4A4,5X,4A2,
(0228)        @      '(LOCAL TIME)'//)
(0229)         WRITE(IRPRT,85)
(0230)      85   FORMAT(//)
```

```
(0231)
(0232)
(0233)  C----locate time log for current station
(0234)    210 DO 240 MINDX=1,KSIZE
(0235)          IF (STNID.EQ.MSID(MINDX)) GOTO 280
(0236)    240   CONTINUE
(0237)
(0238)  C----no current entry, search for hole
(0239)        DO 260 MINDX=1,KSIZE
(0240)          IF (MSID(MINDX).EQ.0) GOTO 270
(0241)    260   CONTINUE
(0242)
(0243)  C----what goes? better abort
(0244)        CALL TNOU('BECAUSE WE HAVE NO ROOM IN LOG-IN ARRAY',39)
(0245)        CALL CLSOUT
(0246)
(0247)
(0248)  C----insert new station name in time log array
(0249)    270 MSID(MINDX) = STNID
(0250)
(0251)    280 ITRY = 0
(0252)    220 ITRY = ITRY+1
(0253)        IF (ITRY.GT.NTRY) GOTO 820
(0254)  C----dial the number if flag-1 not true
(0255)        IF (FLAG(1)) GOTO 320
(0256)        CALL DIAL(ERCODE)
(0257)        IF (ERCODE.NE.0) GOTO 200
(0258)        GOTO 340
(0259)    320 WRITE(1,90)STNID,TPN
(0260)     90         FORMAT(1X,'DIAL TEL. NO. FOR STATION "',A2,
(0261)       $     '"  (',13A2,'), THEN HIT RETURN')
(0262)        CALL PAUS
(0263)  C----get time of linkup
(0264)    340 TIN = TIME$A(TIME)
(0265)  C----request X-TIM systems status
(0266)        CALL SDQUE(ERCODE)
(0267)        IF (ERCODE.NE.0) GOTO(500,830),ERCODE
(0268)  C----word (4) now contains tim status word - save it
(0269)        XSTAT = WORD(4)
(0270)  C----display X-TIM status word
(0271)        CALL OCTAL(XSTAT,CHARAY)
(0272)        WRITE (1,30) CHARAY
(0273)        WRITE(IRPRT,30) CHARAY
(0274)     30   FORMAT(1X'      X-TIM STATUS WORD = ',6A1)
(0275)
(0276)  C*************************************************************************
(0277)  C----If data dump requested,
(0278)  C----check if data RAM is less than half full, no dump if so provided
(0279)  C----no autostart occurred.
(0280)        IF ((AND(XSTAT,:2000).NE.0).OR.(AND(CWORD,:2000).EQ.0)) GOTO 230
(0281)        IF ((WORD(5).GT.:4200).OR.(AND(CWORD,:200).NE.0)) GOTO 230
(0282)        CWORD = AND(CWORD,:75770)
(0283)        WRITE(1,35)
(0284)        WRITE(IRPRT,35)
(0285)     35   FORMAT(1X,'$$TIM RAM LESS THAN HALF FULL, NO DUMP TAKEN$$')
(0286)
```

```
(0287)  C*********************************************************************
(0288)  C----do appropriate dumps and loads as specified by the bits of "CWORD"
(0289)
(0290)  C----get data dump if CWORD(5) or CWORD(8) is set
(0291)    230 ERCODE = 0
(0292)    300 IF (AND(CWORD,:2200).NE.0) CALL XDUMP(KDATA,ERCODE)
(0293)        IF (ERCODE.NE.0) GOTO 500
(0294)
(0295)  C----get program dump if bit CWORD(6) or both bits CWORD(1) and (7) are
(0296)  C---- set
(0297)        ERCODE = 0
(0298)        IF ((AND(CWORD,:1000).NE.0).OR.((AND(CWORD,:40000).NE.0).AND.
(0299)       @     (AND(CWORD,:400).NE.0))) CALL XDUMP(KPROG,ERCODE)
(0300)        IF (ERCODE.NE.0) GOTO 500
(0301)
(0302)  C----transmit program load if bit CWORD(1) is set
(0303)        IF (AND(CWORD,:40000).NE.0) CALL XLOAD(KPROG,ERCODE)
(0304)        IF (ERCODE.NE.0) GOTO 500
(0305)
(0306)  C----transmit scan-rate-list if bit CWORD(2) is set
(0307)        ERCODE = 0
(0308)        IF (AND(CWORD,:20000).NE.0) CALL XLOAD(KSRL,ERCODE)
(0309)        IF (ERCODE.NE.0) GOTO 500
(0310)
(0311)  C----transmit digital-control-list if CWORD(3) is set
(0312)        ERCODE = 0
(0313)        IF (AND(CWORD,:10000).NE.0) CALL XLOAD(KDCL,ERCODE)
(0314)        IF (ERCODE.NE.0) GOTO 500
(0315)
(0316)
(0317)  C----cease communicating with this station
(0318)    420 CALL CEASE(ERCODE)
(0319)        IF (ERCODE.NE.0) GOTO 500
(0320)
(0321)  C----set new time an date only if data RAM is reset
(0322)  C------or if a program load took place
(0323)  C----otherwise retain existing time and date
(0324)        IF ((AND(CWORD,:1).EQ.0).AND.(AND(CWORD,:40000).EQ.0)) GOTO 460
(0325)
(0326)
(0327)  C----store gmt time and julian date of call termination
(0328)        ADATE(MINDX,2) = DOFY$A(TIME)
(0329)        HLDTIM = TIME$A(TIME)+CORR+8.0+KDST
(0330)
(0331)  C----check for overflow of 24 hour
(0332)        IF (HLDTIM.LT.24.0) GOTO 440
(0333)        HLDTIM = HLDTIM-24.0
(0334)        ADATE(MINDX,2) = ADATE(MINDX,2)+.001
(0335)    440 OTIME(MINDX,2) = HLDTIM
(0336)
(0337)    460 CALL DTR$C(OFF)
(0338)  C----get time of breakoff
(0339)        TFIN = TIME$A(TIME)
(0340)  C----compute duration of call and type it
(0341)        IDUR = (TFIN-TIN)*3600.
(0342)        WRITE (1,50)IDUR
```

```
(0343)          WRITE(IRPRT,50) IDUR
(0344)       50   FORMAT (1X, 'DURATION OF CALL = ',I5,' SECS.'///)
(0345)
(0346)    C----put log-in array out to file
(0347)          CALL OPEN@A(A@WRIT+A@SAMF,'X-DATA.LOG.DA',13,4)
(0348)          WRITE(8) (MSID(I),ADATE(I,2),OTIME(I,2),I=1,KSIZE)
(0349)          CALL CLOS@A(4)
(0350)
(0351)          GOTO 200
(0352)
(0353)    C*************************************************************************
(0354)
(0355)    C----if ERCODE = 1, abort this call.
(0356)    C----if ERCODE = 2, abort this station.
(0357)      500 GOTO(540,520),ERCODE
(0358)
(0359)    C----abort station
(0360)      520 WRITE(1,45)STNID
(0361)          WRITE(IRPRT,45)STNID
(0362)       45   FORMAT(1X,'STATION ',A2,' ABORTED')
(0363)          GOTO 840
(0364)
(0365)    C----------retry with current station----------
(0366)      540 WRITE (IRPRT,40) ITRY
(0367)          WRITE(1,40) ITRY
(0368)       40   FORMAT(1X,'TRANSMISSION ATTEMPT #',I1,' FAILED')
(0369)          CALL RTS@C(OFF)
(0370)          CALL DTR@C(OFF)
(0371)          CALL SLEEP@(INTL(5000))
(0372)          GOTO 220
(0373)
(0374)      800 CALL TNOU( 'ERROR IN STATION LIST FILE - ABORT!',35)
(0375)          CALL CLSOUT
(0376)
(0377)      820 WRITE(1,20) STNID
(0378)          WRITE (IRPRT,20) STNID
(0379)       20   FORMAT(1X,'LINKUP DIFFICULTIES WITH TIM STATION - ',
(0380)         1A2,'  THIS STATION ABORTED'///)
(0381)          GOTO 840
(0382)      830 WRITE (IRPRT,60)
(0383)       60   FORMAT(1X,'CALL ABORTED'//)
(0384)          WRITE (1,60)
(0385)    C----hang up tp
(0386)      840 CALL RTS@C(OFF)
(0387)          CALL DTR@C(OFF)
(0388)    C----and wait 5 seconds
(0389)          CALL SLEEP@(INTL(5000))
(0390)          GO TO 200
(0391)
(0392)
(0393)
(0394)
(0395)
(0396)    C----------close out operations----------
(0397)
(0398)      900 WRITE(1,70)
```

```
(0399)          WRITE (IRPRT,70)
(0400)       70   FORMAT(1X,'END OF STATION LIST')
(0401)    C----close station list file
(0402)          IF (CLOS$A(1)) GOTO 910
(0403)          CALL TNOU('COULD NOT CLOSE STATION LIST FILE',33)
(0404)          CALL CLSOUT
(0405)
(0406)      910 IF (CLOS$A(5)) GOTO 920
(0407)          CALL TNOU('COULD NOT CLOSE REPORT FILE',27)
(0408)          CALL CLSOUT
(0409)
(0410)      920 IF (.NOT.FLAG(3)) CALL EXIT
(0411)          WAITTM = (25.00-TIME$A(TIME))*3600.*1000.
(0412)          CALL SLEEP$(WAITTM)
(0413)          GOTO 120
(0414)          END
PROGRAM SIZE:    PROCEDURE - 003236    LINKAGE - 000336    STACK - 690034
0000 ERRORS [<.MAIN.>FTN-REV17.1]
```

```
(0415)         SUBROUTINE PAUS(IARG)
(0416)
(0417)   C*******************************************************************************
(0418)   C                                                                              *
(0419)   C     program pause until return key is pressed.                               *
(0420)   C                                                                              *
(0421)   C*******************************************************************************
(0422)
(0423)
(0424)
(0425)         PRINT 5,IARG
(0426)       5   FORMAT('***PAUSE ',I2)
(0427)         READ(1,10) IPAUS
(0428)      10   FORMAT(1A1)
(0429)         RETURN
(0430)         END
PROGRAM SIZE:   PROCEDURE - 000053    LINKAGE - 000032    STACK - 000030
0000 ERRORS [<PAUS  >FTN-REV17.1]
```

```
(0431)          SUBROUTINE CLSOUT
(0432)
(0433)    C----close out any files that might have been opened,
(0434)    C----put phone on hook, and exit.
(0435)
(0436)    C X-TIM.TALK.DECLAR           /* common variable declaration for X-TIM.TALK
(0436)          NOLIST
(0437)
(0438)          DO 100 I=1,16
(0439)            IF (I.EQ.6) GOTO 100
(0440)            CALL CLOS@A(I)
(0441)      100   CONTINUE
(0442)          CALL RTS@C(OFF)
(0443)          CALL DTR@C(OFF)
(0444)          CALL TNOU('RUN ABORTED',11)
(0445)          CALL EXIT
(0446)          END
PROGRAM SIZE:    PROCEDURE - 000046     LINKAGE - 000040     STACK - 000024
0000 ERRORS [<CLSOUT>FTN-REV17.1]
```

```
(0447)        SUBROUTINE DIAL(ICODE)
(0448)
(0449)  C*******************************************************************************
(0450)  C                                                                              *
(0451)  C                  ****** SUBROUTINE DIAL ******                               *
(0452)  C                                                                              *
(0453)  C     Dial up the number contained in the array "TPN".                         *
(0454)  C     If dialup fails return with nonzero argument.                            *
(0455)  C                                                                              *
(0456)  C*******************************************************************************
(0457)
(0458)
(0459)
(0460)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(0460)        NOLIST
(0461)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0461)        NOLIST
(0462)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0462)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0462)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0462)        NOLIST
(0463)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(0463)  C Insert file for mnemonic APPLIB keys (FTN)
(0463)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(0463)        NOLIST
(0464)
(0465)
(0466)        INTEGER CHAR
(0467)
(0468)        ICODE = 0
(0469)
(0470)  C----we will try to link up 5 times only!
(0471)        DO 300 ITRY=1,5
(0472)  C----turn o "DTR"
(0473)        CALL DTR$C(ON)
(0474)  C----wait for "dsr"
(0475)    200 IF(.NOT.DSR$C(0)) GOTO 200
(0476)  C----dial the number
(0477)  C----turn on transmitter
(0478)        CALL RTS$C(ON)
(0479)  C----wait for CTS
(0480)    210 IF (.NOT.CTS$C(0)) GOTO 210
(0481)        CALL XMTCH(STX)
(0482)
(0483)  C----must pick up a space char. before end of array
(0484)        DO 220 I=1,12
(0485)          CALL XMTCH(IBYSW(TPN(I)))
(0486)          IF (TPN(I+1).EQ.'  ') GOTO 240
(0487)    220   CONTINUE
(0488)
(0489)  C----bad number - abort dialup
(0490)        CALL XMTCH(SOH)
(0491)        WRITE(1,15)
(0492)        WRITE (1RPRT,15)
(0493)     15   FORMAT(1X,'BAD TP.# - CALL ABORTED!')
(0494)        GOTO 900
```

```
(0495)
(0496)  C----clean out in buffer
(0497)    240 CALL MTBF$C(COMLIN)
(0498)        CALL XMTCH(S1)
(0499)        CALL XMTCH(ETX)
(0500)  C----turn off transmitter
(0501)        CALL RTS$C(OFF)
(0502)
(0503)  C----pick up response character
(0504)        CHAR = MODIN(90)
(0505)        IF (CHAR.EQ.-1) GOTO 360
(0506)  C----convert to literal format
(0507)        CHAR = OR(LS(CHAR,8),:100240)
(0508)        IF (CHAR.EQ.'A') GOTO 340
(0509)        IF (CHAR.EQ.'B') GOTO 280
(0510)
(0511)        WRITE (1,20) CHAR
(0512)        WRITE (IRPRT,20) CHAR
(0513)     20 FORMAT (1X'DIAL ERROR: RESPONSE CHARACTER IS - 'A1)
(0514)        GOTO 290
(0515)
(0516)    280 WRITE(1,30) STNID
(0517)        WRITE (IRPRT,30) STNID
(0518)     30 FORMAT(1X'LINKUP NOT MADE TO STATION - ',A2)
(0519)  C----hang up modem and try again after 10 seconds
(0520)    290 CALL DTR$C(OFF)
(0521)        CALL SLEEP$(0010000)
(0522)    300 CONTINUE
(0523)
(0524)  C----unable to link up, abort this station
(0525)    900 CALL DTR$C(OFF)
(0526)        WRITE(IRPRT,45)
(0527)     45   FORMAT(1H0/1H0)
(0528)        ICODE = 1
(0529)        RETURN
(0530)
(0531)
(0532)    340 WRITE (IRPRT,40) STNID
(0533)        WRITE(1,40) STNID
(0534)     40 FORMAT (1X'LINK UP TO STATION - ',A2,'  SUCCESSFUL')
(0535)        RETURN
(0536)
(0537)  C----rubout character means hardware problems.  stop everything.
(0538)    320 WRITE (1,25)
(0539)        WRITE (IRPRT,25)
(0540)     25   FORMAT(1X, 'PARITY ERROR IN RESPONSE CHAR. - ABORT RUN!')
(0541)        GOTO 380
(0542)    360 WRITE (1,35)
(0543)        WRITE (IRPRT,35)
(0544)     35   FORMAT(1X, 'NO RESPONSE FROM MODEM DIALER - ABORT RUN!')
(0545)    380 CALL DTR$C(OFF)
(0546)        CALL EXIT
(0547)
(0548)        END
PROGRAM SIZE:   PROCEDURE - 000634    LINKAGE - 000116    STACK - 000034
0000 ERRORS [<DIAL  >FTN-REV17.1]
```

```
(0549)        SUBROUTINE XMTCH(CHAR)
(0550)
(0551)  C*****************************************************************************
(0552)  C                                                                            *
(0553)  C                    ****** XMTCH ******                                     *
(0554)  C                                                                            *
(0555)  C     subroutine for character by character transmission                     *
(0556)  C     character to be transmitted contained in low order byte.               *
(0557)  C                                                                            *
(0558)  C*****************************************************************************
(0559)
(0560)
(0561)
(0562)
(0563)
(0564)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(0564)        NOLIST
(0565)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0565)        NOLIST
(0566)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0566)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0566)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0566)        NOLIST
(0567)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(0567)  C Insert file for mnemonic APPLIB keys (FTN)
(0567)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(0567)        NOLIST
(0568)
(0569)        INTEGER CHAR,ICHAR(6),STAT(2)
(0570)
(0571)  C----print out character if FLAG(7) is true
(0572)        IF (.NOT.FLAG(7)) GOTO 200
(0573)        CALL OCTAL(AND(CHAR,:377),ICHAR)
(0574)        PRINT 10,CHAR,ICHAR
(0575)     10   FORMAT(1A2,2X,6A1)
(0576)    200 ICHAR(1) = IBYSW(CHAR)
(0577)        CALL T$AMLC(COMLIN,LOC(ICHAR),1,3,STAT)
(0578)        RETURN
(0579)        END
PROGRAM SIZE:   PROCEDURE - 000101    LINKAGE - 000054    STACK - 000334
0000 ERRORS (<XMTCH >FTN-REV17.1)
```

```
(0580)        SUBROUTINE XMTLB(CHAR)
(0581)
(0582)
(0583)  C*****************************************************************************
(0584)  C                                                                            *
(0585)  C      ****** SUBROUTINE XMTLB ******                                        *
(0586)  C                                                                            *
(0587)  C     routine to receive a string of characters sandwiched                   *
(0588)  C     between "SYNC" & "EOM" characters and stored in an array.              *
(0589)  C     when the "EOM" is received the carrier is turned on and                *
(0590)  C     the character array is transmitted.                                    *
(0591)  C                                                                            *
(0592)  C*****************************************************************************
(0593)
(0594)
(0595)
(0596)
(0597)  C X-TIM.TALK.DECLAR           /* common variable declaration for X-TIM.TALK
(0597)        NOLIST
(0598)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0598)        NOLIST
(0599)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0599)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0599)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0599)        NOLIST
(0600)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(0600)  C Insert file for mnemonic APPLIB keys (FTN)
(0600)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(0600)        NOLIST
(0601)
(0602)        INTEGER CHAR,ARROUT(200),STAT(2),AMLCBF(40)
(0603)        LOGICAL FIN
(0604)
(0605)  C----print "CHAR" in octal if FLAG-2 is true
(0606)        IF (FLAG(8)) CALL OCTPRT(CHAR)
(0607)
(0608)  C----check if first character, if so initialize character counter
(0609)  C------such that 1 null characte will be sent initially
(0610)        IF (CHAR.NE.SYNC)  GOTO 180
(0611)        N = 1
(0612)        ARROUT(1) = 0
(0613)  C----update array index
(0614)    180 J = N/2+1
(0615)  C----left or right byte
(0616)        IF (MOD(N,2).EQ.0) ARROUT(J) = LS(CHAR,8)
(0617)        IF (MOD(N,2).NE.0) ARROUT(J) = OR(ARROUT(J),CHAR)
(0618)        N = N+1
(0619)  C----if last character, start transmitting
(0620)        IF (CHAR.EQ.EOM) GOTO 200
(0621)  C----update character count
(0622)        RETURN
(0623)
(0624)    200   CALL TIMLT(15,ERCODE)
(0625)
(0626)  C----be sure carrier is not on
(0627)    205   IF (.NOT.CD$C(0)) GOTO 210
```

```
(0628)          CALL TIMLT(0,ERCODE)
(0629)        IF (ERCODE.NE.0) GOTO 900
(0630)          GOTO 205
(0631)
(0632)  C----turn on transmitter
(0633)    210 CALL RTS$C(ON)
(0634)  C----be sure 'CTS' is on but don't wait forever
(0635)        CALL TIMLT(5,ERCODE)
(0636)    220 IF (CTS$C(0)) GOTO 230
(0637)        CALL TIMLT(0,ERCODE)
(0638)        IF (ERCODE.EQ.0) GOTO 220
(0639)        PRINT 10
(0640)        WRITE(IRPRT,10)
(0641)     10   FORMAT(1X,'TIMEOUT BEFORE 'CTS' CAME ON!')
(0642)        CALL CLSOUT
(0643)
(0644)  C*********************************************************************
(0645)  C
(0646)  C----transmit in blocks of 80 characters if necessary
(0647)
(0648)  C----initialize array pointer
(0649)    230 JJ = 1
(0650)        FIN = .FALSE.
(0651)        CALL SLEEP$(INTL(300))
(0652)  C----determine # of characters to be transmitted in this round
(0653)    250 IREM = N-JJ+1
(0654)
(0655)  C----this will be the last round
(0656)        IF (IREM.GT.80) GOTO 270
(0657)        K = IREM
(0658)        FIN = .TRUE.
(0659)        GOTO 240
(0660)
(0661)    270 K = 80
(0662)
(0663)  C----load the transmission buffer
(0664)    240 KWORDS = (K+1)/2
(0665)        DO 290 M=1,KWORDS
(0666)          AMLCBF(M) = ARROUT(JJ/2+M)
(0667)    290   CONTINUE
(0668)
(0669)
(0670)
(0671)  C----start transmitting
(0672)    260 CALL T$AMLC(COMLIN,LOC(AMLCBF),K,S,STAT)
(0673)  C----compute transmission time
(0674)        WAITTM = (FLOAT(K*11)*1000.)/1200.
(0675)  C----and sleep while it takes place.
(0676)        CALL SLEEP$(WAITTM)
(0677)        IF (.NOT.FLAG(4)) GOTO 280
(0678)        PRINT 20,K,WAITTM
(0679)     20   FORMAT('K = ',I3/'WAITTM = ',I4)
(0680)        DO 400 I=1,KWORDS
(0681)          CALL OCTPRT(RS(AMLCBF(I),8))
(0682)          CALL OCTPRT(RT(AMLCBF(I),8))
(0683)        CALL TNOUA('  ',2)
```

```
(0684)          CALL OCTPRT(AMLCBF(I))
(0685)      400   CONTINUE
(0686)
(0687)    C*****************************************************************
(0688)
(0689)
(0690)    C----wait for empty buffer
(0691)      280 CALL T@AMLC(COMLIN,LOC(AMLCBF),80,5,STAT)
(0692)          IF (STAT(1).NE.0) GOTO 300
(0693)          CALL SLEEP@(INTL(100))
(0694)          GOTO 280
(0695)
(0696)    C----are we finished?
(0697)      300 JJ = JJ+80
(0698)          IF (.NOT.FIN) GOTO 250
(0699)
(0700)    C----and a bit more
(0701)          CALL SLEEP@(INTL(100))
(0702)    C----turn off carrier
(0703)          CALL RTS@C(OFF)
(0704)          RETURN
(0705)
(0706)
(0707)      900 CALL TNOU('CARRIER STAYED ON TOO LONG!',27)
(0708)          CALL CLSOUT
(0709)
(0710)
(0711)          END
PROGRAM SIZE:   PROCEDURE - 000554    LINKAGE - 000504    STACK - 000034
0000 ERRORS [<XMTLB >FTN-REV17.1]
```

```
(0712)        BLOCK DATA
(0713)
(0714)  C*******************************************************************************
(0715)  C                                                                              *
(0716)  C     ****** X:BLKDATA ******                                                  *
(0717)  C                                                                              *
(0718)  C    initialize values used in X-TIM.TALK                                      *
(0719)  C                                                                              *
(0720)  C*******************************************************************************
(0721)
(0722)
(0723)
(0724)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(0724)        NOLIST
(0725)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0725)        NOLIST
(0726)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0726)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0726)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0726)        NOLIST
(0727)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(0727)  C Insert file for mnemonic APPLIB keys (FTN)
(0727)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(0727)        NOLIST
(0728)
(0729)
(0730)        DATA TPN/13*0/
(0731)
(0732)        DATA STX,SI,ETX,SOH
(0733)       1  /:002,:017,:003,:001/
(0734)
(0735)        DATA SDATA,SPROC,SSTAT,LPROC,LPRCC,LSRL,LDCL,TERM
(0736)       1  /:140,:141,:142,:143,:144,:145,:146,:147/
(0737)
(0738)        DATA ACK,NAK,EOM,SYNC/:106,:125,:103,:123/
(0739)        DATA IRPRT/9/
(0740)        DATA PSFIL/'CROSS>.TIM.>   .LOGR*BN'/
(0741)        DATA DSFIL/'CROSS>.TIM.>   .DCL*BN'/
(0742)        DATA SSFIL/'CROSS>.TIM.>   .SRL*BN'/
(0743)        DATA PDFIL/'   .PDUMP*DA'/
(0744)        DATA DDFIL/'   .DDUMP*DA'/
(0745)        DATA FLAG/9*.FALSE./
(0746)        DATA ON,OFF/.TRUE.,.FALSE./
(0747)        DATA CTLLIN,COMLIN/:34,:35/
(0748)
(0749)        END
PROGRAM SIZE:   PROCEDURE - 000000    LINKAGE - 000000    STACK - 000000
0000 ERRORS [<.DATA.>FTN-REV17.1]
```

```
(0750)        SUBROUTINE SDQUE( ICODE)
(0751)
(0752)
(0753)  C*****************************************************************************
(0754)  C                                                                            *
(0755)  C      ****** SUBROUTINE SDQUE ******                                        *
(0756)  C                                                                            *
(0757)  C     send a message packet to X-TIM                                         *
(0758)  C     transmit tim status to data develop. format of message packet          *
(0759)  C     as follows:                                                            *
(0760)  C                    variable value                                          *
(0761)  C      sync code       SYNC   123                                            *
(0762)  C      packet size     PKSIZ    6                                            *
(0763)  C      stat. id.       STNID                                                 *
(0764)  C      op code         SSTAT  142                                            *
(0765)  C      checksum        CKSUM                                                 *
(0766)  C      end-of-mess.    EOM    103                                            *
(0767)  C                                                                            *
(0768)  C     ICODE = 1 if tp call is to be aborted.                                 *
(0769)  C     ICODE = 2 if station is to be aborted.                                 *
(0770)  C                                                                            *
(0771)  C*****************************************************************************
(0772)
(0773)
(0774)  C X-TIM.TALK.DECLAR           /* common variable declaration for X-TIM.TALK
(0774)        NOLIST
(0775)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0775)        NOLIST
(0776)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0776)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0776)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0776)        NOLIST
(0777)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(0777)  C Insert file for mnemonic APPLIB keys (FTN)
(0777)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(0777)        NOLIST
(0778)
(0779)
(0780)        INTEGER*2 CHARAY(6),HOURS,MINS,SECS
(0781)
(0782)        PKSIZ = 6
(0783)        ICODE = 0
(0784)
(0785)  C----we shall limit our attempts to 3 for now
(0786)        ITRY = 0
(0787)    200 CALL SLEEP$(000001000)
(0788)        ITRY = ITRY+1
(0789)        IF (ITRY.GT.NTRY) GOTO 900
(0790)        CALL TIMLT(15,ERCODE)
(0791)
(0792)  C*************************************************
(0793)  C----transmit system status request message
(0794)
(0795)    240 CKSUM = 0
(0796)  C----transmit sync char.
(0797)        CALL XMTLB(SYNC)
```

```
(0798)
(0799)  C----transmit packet size in 2 - 6-bit bytes,
(0800)  C----highorder first
(0801)        TEMP = AND(RS(PKSIZ,6),:77)
(0802)        CKSUM = CKSUM+TEMP
(0803)        CALL XMTLB(TEMP)
(0804)        TEMP = AND(PKSIZ,:77)
(0805)        CKSUM = CKSUM+TEMP
(0806)        CALL XMTLB(TEMP)
(0807)
(0808)  C----transmit station id in 2 - 6-bit bytes
(0809)        TEMP = AND(IBYSW(STNID),:77)
(0810)        CKSUM = CKSUM+TEMP
(0811)        CALL XMTLB(TEMP)
(0812)        TEMP = AND(STNID,:77)
(0813)        CKSUM = CKSUM+TEMP
(0814)        CALL XMTLB(TEMP)
(0815)
(0816)  C----transmit system status op code
(0817)        CKSUM = CKSUM+SSTAT
(0818)        CALL XMTLB(SSTAT)
(0819)
(0820)  C----send checksum in 2 6-bit bytes
(0821)        TEMP = RT(RS(CKSUM,6),6)
(0822)        CALL XMTLB(TEMP)
(0823)        TEMP = RT(CKSUM,6)
(0824)        CALL XMTLB(TEMP)
(0825)
(0826)  C----and the end-of-message char.
(0827)        CALL XMTLB(EOM)
(0828)
(0829)
(0830)
(0831)  C*************************************************************
(0832)  C----receive system status message
(0833)
(0834)
(0835)  C----wait for sync code
(0836)    320 TEMPL = MODIN(MWT)
(0837)        IF (TEMPL.EQ.-1) GOTO 200
(0838)        IF (TEMPL.NE.SYNC) GOTO 320
(0839)        TEMPL = MODIN(MWT)
(0840)        IF (TEMPL.EQ.-1) GOTO 200
(0841)  C----check if "nak".
(0842)  C----if not, should be a system-status packet incoming.
(0843)  C----if "nak" attempt another transmission
(0844)        IF (TEMPL.NE.NAK) GOTO 330
(0845)  C----"nak" received, tell about it.
(0846)        WRITE(1,41)
(0847)        WRITE(IRPRT,41)
(0848)     41   FORMAT(1X'BAD OUTGOING TRANSMISSION')
(0849)        GOTO 200
(0850)
(0851)  C----should be station id bytes -
(0852)  C---- convert from 6-bit stripped ascii to 7-bit ascii
(0853)    330 TEMPH = TEMPL+:300
```

```
(0854)         TEMPL = MODIN(MWT)
(0855)         IF (TEMPL.EQ.-1) GOTO 200
(0856)         WORD(I) = LS(TEMPH,8)+TEMPL+:300
(0857)
(0858)
(0859)         I = 2
(0860)     340 TEMPH = MODIN(MWT)
(0861)         IF (TEMPH.EQ.-1) GOTO 200
(0862)         IF (TEMPH.EQ.EOM) GOTO 345
(0863)         TEMPL = MODIN(MWT)
(0864)         IF (TEMPL.EQ.-1) GOTO 200
(0865)         IF (TEMPL.EQ.EOM) GOTO 345
(0866)         WORD(I) = LS(TEMPH,6)+TEMPL
(0867)         I = I+1
(0868)         GOTO 340
(0869)
(0870)
(0871)   C----at this point the data array should contain:
(0872)   C----   word(1) = station id
(0873)   C----   word(2) = clock reg. hi order
(0874)   C----   word(3) = clock reg. lo order
(0875)   C----   word(4) = X-TIM system status
(0876)   C----   word(5) = data block pointer
(0877)   C----   word(6) = checksum
(0878)
(0879)   C----print the array and intermediate checksum values if flag-6 is true.
(0880)     345 IF (.NOT.FLAG(6)) GOTO 352
(0881)         DO 350 J=1,I
(0882)           PRINT 12,J
(0883)      12     FORMAT(I2)
(0884)           CALL TNOUA('  ',2)
(0885)           CALL OCTPRT(WORD(J))
(0886)     350   CONTINUE
(0887)     352 CKSUM = AND(IBYSW(WORD(1)),:77)+AND(WORD(1),:77)
(0888)         IF (FLAG(6)) CALL OCTPRT(CKSUM)
(0889)         DO 360 I=2,5
(0890)           CKSUM = CKSUM+RS(WORD(I),6)+AND(WORD(I),:77)
(0891)           IF (FLAG(6)) CALL OCTPRT(CKSUM)
(0892)     360   CONTINUE
(0893)         IF (FLAG(6)) CALL OCTPRT(CKSUM)
(0894)   C----if a hack checksum compare, go for new transmission
(0895)         IF (CKSUM.EQ.WORD(6)) GOTO 370
(0896)   C----received back packet, tell about it.
(0897)         PRINT 42
(0898)         WRITE(IRPRT,42)
(0899)      42   FORMAT(1X,'BAD INCOMING TRANSMISSION (CHECKSUM ERROR).')
(0900)         GOTO 200
(0901)
(0902)   C----correct station id?
(0903)     370 IF (NTRID.NE.WORD(1)) GOTO 500
(0904)         WRITE(1,10) WORD(1)
(0905)         WRITE(IRPRT,10) WORD(1)
(0906)      10   FORMAT(1X,'STATION REPORTING - ',A2)
(0907)
(0908)         IF (AND(WORD(4),:4000).EQ.0) GOTO 380
(0909)         PRINT 20
```

```
(0910)         WRITE (IRPRT,20)
(0911)      20   FORMAT(1X,'***CAUTION*** - A MANUAL RESET HAS OCCURRED SINCE ',
(0912)     $     'LAST INTERROGATION!')
(0913)
(0914)   C----show the system status
(0915)     380 IF (AND(WORD(4),:2000).EQ.0) GOTO 420
(0916)         PRINT 22
(0917)         WRITE(IRPRT,22)
(0918)      22   FORMAT(1X,'**ATTENTION** - AUTOSTART SINCE LAST INTERROGATION')
(0919)         IF(AND(CWORD,:4000).EQ.0) GOTO 440
(0920)         WRITE(IRPRT,23)
(0921)      23   FORMAT(1X'     A PROGRAM RELOAD WILL TAKE PLACE')
(0922)         PRINT 23
(0923)   C----set data dump, program dump, and program load switches
(0924)         CWORD = OR(CWORD,:43000)
(0925)         GOTO 440
(0926)     420 PRINT 24
(0927)         WRITE(IRPRT,24)
(0928)      24   FORMAT(1X,'NO AUTO RESTARTS SINCE LAST INTERROGATION')
(0929)     440 IF (AND(WORD(4),:40).NE.0) GOTO 460
(0930)         WRITE (IRPRT,26)
(0931)      26   FORMAT(1X,'!!!AUTO RESTART SWITCH NOT ACTIVE!!!')
(0932)         PRINT 26
(0933)   C----show current relative time of X-TIM since last clock reset
(0934)     460 IF (AND(WORD(4),:200).EQ.0) GOTO 520
(0935)         WRITE (1,32)
(0936)         WRITE (IRPRT,32)
(0937)      32   FORMAT(1X'******ATTENTION - BATTERY VOLTAGE LOW OR POWER ',
(0938)     $     'FAILED******')
(0939)     520 IF (AND(WORD(4),:100).EQ.0) GOTO 540
(0940)         WRITE(1,34)
(0941)         WRITE(IRPRT,34)
(0942)      34   FORMAT(1X,'!!!ATTENTION - DATA RAM OVERFLOWED!!!')
(0943)     540 XTIME = FLOAT(WORD(2))*4096.+FLOAT(WORD(3))
(0944)         HOURS = XTIME/3600.
(0945)         MINS = AMOD(XTIME,3600.)/60.
(0946)         SECS = AMOD(XTIME,60.)
(0947)         WRITE(1,14)HOURS,MINS,SECS
(0948)         WRITE (IRPRT,14) HOURS,MINS,SECS
(0949)      14 FORMAT(1X,' ELAPSE TIME SINCE LAST CLOCK RESET - '
(0950)     $     ,I3,' HOURS',2X12,' MINUTES',2X12,' SECONDS')
(0951)         CALL OCTAL(WORD(5),CHARAY)
(0952)         WRITE(1,50)CHARAY
(0953)         WRITE(IRPRT,50)CHARAY
(0954)      50   FORMAT(1X'DATA BLOCK POINTER = ',6A1)
(0955)
(0956)         RETURN
(0957)
(0958)   C----abort-return
(0959)     600 WRITE (1,30) WORD(1),NTRID
(0960)         WRITE (IRPRT,30) WORD(1),NTRID
(0961)      30   FORMAT(1X,'WRONG STATION, GOT ',A2,'     BUT EXPECT ',A2,'!')
(0962)   C----cause this station to terminate linkup
(0963)         NTRID = WORD(1)
(0964)         CWORD = 0
(0965)         CALL CEASE(ENCODE)
```

```
(0966)          IF (ERCODE.NE.0) GOTO 900
(0967)          ICODE = 2
(0968)          RETURN
(0969)      900 ICODE = 1
(0970)          RETURN
(0971)
(0972)          END
PROGRAM SIZE:    PROCEDURE - 002010     LINKAGE - 000152     STACK - 000036
0000 ERRORS [<SDQUE >FTN-REV17.1]
```

```
(0973)        SUBROUTINE SDUMY( ICODE)
(0974)
(0975)
(0976)  C*****************************************************************************
(0977)  C                                                                            *
(0978)  C     ****** SUBROUTINE SDUMY ******                                         *
(0979)  C                                                                            *
(0980)  C    send a dummy message packet to X-TIM                                    *
(0981)  C    transmit tim status to data develop. format of message packet           *
(0982)  C    as follows:                                                             *
(0983)  C                    variable value                                          *
(0984)  C     sync code      SYNC   123                                              *
(0985)  C     packet size    PKSIZ    6                                             *
(0986)  C     stat. id.      STNID                                                   *
(0987)  C     op code        SSTAT  142                                              *
(0988)  C     checksum       CKSUM                                                   *
(0989)  C     end-of-mess.   EOM    103                                             *
(0990)  C                                                                            *
(0991)  C    ICODE = 1 if tp call is to be aborted.                                  *
(0992)  C    ICODE = 2 if station is to be aborted.                                  *
(0993)  C                                                                            *
(0994)  C*****************************************************************************
(0995)
(0996)
(0997)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(0997)        NOLIST
(0998)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(0998)        NOLIST
(0999)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(0999)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(0999)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(0999)        NOLIST
(1000)  C ASKEYS, APPLIB>SOURCE, ELS, 02/12/79
(1000)  C Insert file for mnemonic APPLIB keys (FTN)
(1000)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(1000)        NOLIST
(1001)
(1002)
(1003)        INTEGER*2 CHARAY(6),HOURS,MINS,SECS
(1004)
(1005)        PKSIZ = 6
(1006)        ICODE = 0
(1007)
(1008)  C----we shall limit our attempts to 3 for now
(1009)        ITRY = 0
(1010)    200 CALL SLEEP0(INTL(1000))
(1011)        ITRY = ITRY+1
(1012)        IF (ITRY.GT.NTRY) GOTO 900
(1013)        CALL TIMLT(15,ERCODE)
(1014)
(1015)  C*****************************************************
(1016)  C----transmit system status request message
(1017)
(1018)    240 CKSUM = 0
(1019)  C----transmit sync char.
(1020)        CALL XMTLB(SYNC)
```

```
(1021)
(1022)  C----transmit packet size in 2 - 6-bit bytes,
(1023)  C----highorder first
(1024)        TEMP = AND(RS(PKSIZ,6),:77)
(1025)        CKSUM = CKSUM+TEMP
(1026)        CALL XMTLB(TEMP)
(1027)        TEMP = AND(PKSIZ,:77)
(1028)        CKSUM = CKSUM+TEMP
(1029)        CALL XMTLB(TEMP)
(1030)
(1031)  C----transmit station id in 2 - 6-bit bytes
(1032)        TEMP = AND(IBYSW(STNID),:77)
(1033)        CKSUM = CKSUM+TEMP
(1034)        CALL XMTLB(TEMP)
(1035)        TEMP = AND(STNID,:77)
(1036)        CKSUM = CKSUM+TEMP
(1037)        CALL XMTLB(TEMP)
(1038)
(1039)  C----transmit system status op code
(1040)        CKSUM = CKSUM+SSTAT
(1041)        CALL XMTLB(SSTAT)
(1042)
(1043)  C----send checksum in 2 6-bit bytes
(1044)        TEMP = RT(RS(CKSUM,6),6)
(1045)        CALL XMTLB(TEMP)
(1046)        TEMP = RT(CKSUM,6)
(1047)        CALL XMTLB(TEMP)
(1048)
(1049)  C----and the end-of-message char.
(1050)        CALL XMTLB(EOM)
(1051)
(1052)
(1053)
(1054)  C************************************************************
(1055)  C----receive system status message
(1056)
(1057)
(1058)  C----wait for sync code
(1059)    320 TEMPL = MODIN(MWT)
(1060)        IF (TEMPL.EQ.-1) GOTO 200
(1061)        IF (TEMPL.NE.SYNC) GOTO 320
(1062)        TEMPL = MODIN(MWT)
(1063)        IF (TEMPL.EQ.-1) GOTO 200
(1064)  C----check if 'nak'
(1065)  C----if not, should be a system-status packet incoming
(1066)  C----if 'nak' attempt another transmission
(1067)        IF (TEMPL.NE.NAK) GOTO 330
(1068)  C----'nak' received, tell about it.
(1069)        WRITE(1,41)
(1070)        WRITE(INPRT,41)
(1071)     41   FORMAT(1X'BAD OUTCOING TRANSMISSION')
(1072)        GOTO 200
(1073)
(1074)  C----should be station id bytes -
(1075)  C----  convert from 6-bit stripped ascii to 7-bit ascii
(1076)    330 TEMPH = TEMPL+300
```

```
(1077)          TEMPL = MODIN(MWT)
(1078)          IF (TEMPL.EQ.-1) GOTO 200
(1079)          WORD(1) = LS(TEMPH,8)+TEMPL+:300
(1080)
(1081)
(1082)          I = 2
(1083)      340 TEMPH = MODIN(MWT)
(1084)          IF (TEMPH.EQ.-1) GOTO 200
(1085)          IF (TEMPH.EQ.EOM) GOTO 345
(1086)          TEMPL = MODIN(MWT)
(1087)          IF (TEMPL.EQ.-1) GOTO 200
(1088)          IF (TEMPL.EQ.EOM) GOTO 345
(1089)          WORD(I) = LS(TEMPH,6)+TEMPL
(1090)          I = I+1
(1091)          GOTO 340
(1092)
(1093)
(1094)    C----at this point the data array should contain:
(1095)    C----   WORD(1) = station id
(1096)    C----   WORD(2) = clock reg. hi order
(1097)    C----   WORD(3) = clock reg. lo order
(1098)    C----   WORD(4) = X-TIM system status
(1099)    C----   WORD(5) = data block pointer
(1100)    C----   WORD(6) = checksum
(1101)    C----but we throw it away
(1102)      345 RETURN
(1103)
(1104)      900 ICODE = 1
(1105)          RETURN
(1106)          END
PROGRAM SIZE:    PROCEDURE - 000330    LINKAGE - 000110    STACK - 000030
0000 ERRORS ((SDUMY )FTN-REV17.1)
```

```
(1107)        INTEGER FUNCTION MODIN(IARG)
(1108)
(1109)  C*******************************************************************************
(1110)  C                                                                              *
(1111)  C     ****** FUNCTION SUBPROGRAM MODIN ******                                  *
(1112)  C                                                                              *
(1113)  C     function routine to fetch contents of                                    *
(1114)  C     modem receiver when available.                                           *
(1115)  C     character returned in low byte.                                          *
(1116)  C     If no character arrives within [IARG] seconds                            *
(1117)  C     a return is taken with the function = -1.                                *
(1118)  C                                                                              *
(1119)  C*******************************************************************************
(1120)
(1121)
(1122)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(1122)        NOLIST
(1123)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(1123)        NOLIST
(1124)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(1124)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(1124)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(1124)        NOLIST
(1125)  C A*KEYS, APPLIB>SOURCE, ELS, 02/12/79
(1125)  C Insert file for mnemonic APPLIB keys (FTN)
(1125)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(1125)        NOLIST
(1126)
(1127)        INTEGER CHAR(1),STAT(2),ERCODE
(1128)
(1129)        CALL TIMLT(IARG,ERCODE)
(1130)    200 CALL T$AMLC(COMLIN,LOC(CHAR),1,4,STAT)
(1131)        IF (STAT(1).NE.0) GOTO 220
(1132)        CALL TIMLT(0,ERCODE)
(1133)        IF (ERCODE.NE.0) GOTO 240
(1134)        CALL SLEEP$(INTL(500))
(1135)        GOTO 200
(1136)
(1137)    220 CALL T$AMLC(COMLIN,LOC(CHAR),1,1,STAT)
(1138)        MODIN = RS(CHAR(1),8)
(1139)        IF (FLAG(5)) CALL OCTPRT(RS(CHAR(1),8))
(1140)        RETURN
(1141)
(1142)    240 MODIN = -1
(1143)        RETURN
(1144)        END
PROGRAM SIZE:   PROCEDURE - 000114    LINKAGE - 000046    STACK - 000034
0000 ERRORS [<MODIN >FTN-REV17.1]
```

```
(1145)        SUBROUTINE TIMLT(ARG,ICODE)
(1146)
(1147)  C*****************************************************************************
(1148)  C                                                                            *
(1149)  C     ****** SUBROUTINE TIMLT ******                                         *
(1150)  C                                                                            *
(1151)  C    time out routine. if "ARG" nonzero, a counter is                        *
(1152)  C    initialize. all subsequent call with "ARG" = 0 will                     *
(1153)  C    return immediately and normally with ICODE = 0 unless                   *
(1154)  C    it reaches the initialized value, then the normal                       *
(1155)  C    return is taken with ICODE = 1.                                         *
(1156)  C                                                                            *
(1157)  C*****************************************************************************
(1158)
(1159)
(1160)
(1161)        REAL*4 H,M,S,TI
(1162)        REAL*8 TIME(1)
(1163)        INTEGER ARG
(1164)        COMMON /HTIME/TOUT      /* CANNOT BE A STACK VARIABLE
(1165)
(1166)        ICODE = 0
(1167)
(1168)        CALL TIME$A(TIME)
(1169)        DECODE(8,20,TIME)H,M,S
(1170)     20   FORMAT(F2.0,1X,F2.0,1X,F2.0)
(1171)        TI = 3600.*H+60.*M+S
(1172)
(1173)        IF(ARG.NE.0) GOTO 100
(1174)
(1175)        IF (TI.GE.TOUT)ICODE = 1
(1176)        RETURN
(1177)
(1178)    100 TOUT = TI+FLOAT(ARG)
(1179)        RETURN
(1180)
(1181)        END
PROGRAM SIZE:   PROCEDURE - 000132    LINKAGE - 000050    STACK - 000034
0000 ERRORS [<TIMLT >FTN-REV17.1]
```

```
(1182)        SUBROUTINE CEASE( ICODE)
(1183)
(1184)
(1185)  C*****************************************************************************
(1186)  C                                                                            *
(1187)  C     ****** SUBROUTINE CEASE ******                                         *
(1188)  C                                                                            *
(1189)  C    send a "cease communication" message to X-TIM                           *
(1190)  C     format of message packet as follows:                                   *
(1191)  C                  variable value(octal)                                     *
(1192)  C     sync code      SYNC   123                                             *
(1193)  C     packet size    PKSIZ   11                                             *
(1194)  C     stat. id.      STNID                                                  *
(1195)  C     op code        TERM   147                                             *
(1196)  C     # secs. in                                                            *
(1197)  C       curr. min.                                                         *
(1198)  C     action word    AWORD                                                  *
(1199)  C     checksum       CKSUM                                                  *
(1200)  C     end-of-mess.   EOM    103                                            *
(1201)  C                                                                            *
(1202)  C    abnormal return is taken if tp call is to be aborted.                   *
(1203)  C                                                                            *
(1204)  C*****************************************************************************
(1205)
(1206)
(1207)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(1207)        NOLIST
(1208)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(1208)        NOLIST
(1209)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(1209)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(1209)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(1209)        NOLIST
(1210)  C A*KEYS, APPLIB>SOURCE, ELS, 02/12/79
(1210)  C Insert file for mnemonic APPLIB keys (FTN)
(1210)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(1210)        NOLIST
(1211)
(1212)
(1213)        INTEGER*2 AWORD
(1214)        REAL*8 ATIME
(1215)
(1216)        PKSIZ = 9
(1217)        ITRY = 0
(1218)    200 ITRY = ITRY+1
(1219)        IF (ITRY.GT.NTRY) GOTO 900
(1220)
(1221)  C*************************************************************
(1222)  C----transmit termination message
(1223)
(1224)  C----get decimal time
(1225)    240 ATIME = (TIME$A(DATE)+CORR)*60.
(1226)  C----compute seconds component
(1227)        NSEC = (ATIME-DINT(ATIME))*60.
(1228)  C----if TIM clock regs. reset, wait for allowable time window
(1229)        IF (AND(CWORD,:2).EQ.0) GOTO 250
```

```
(1230)  C----check if within disallowed time window
(1231)        IF ((NSEC.GE.20).AND.(NSEC.LE.40)) GOTO 250
(1232)  C----mark time with dummy messages if so
(1233)        CALL SDUMY(ERCODE)
(1234)        GOTO 240
(1235)
(1236)    250 CKSUM = 0
(1237)  C----transmit sync char.
(1238)        CALL XMTLB(SYNC)
(1239)
(1240)  C----transmit packet size in 2 - 6-bit bytes,
(1241)  C----highorder first
(1242)        TEMP = AND(RS(PKSIZ,6),:77)
(1243)        CKSUM = CKSUM+TEMP
(1244)        CALL XMTLB(TEMP)
(1245)        TEMP = AND(PKSIZ,:77)
(1246)        CKSUM = CKSUM+TEMP
(1247)        CALL XMTLB(TEMP)
(1248)
(1249)  C----transmit station id in 2 - 6-bit bytes
(1250)        TEMP = AND(IBYSW(STNID),:77)
(1251)        CKSUM = CKSUM+TEMP
(1252)        CALL XMTLB(TEMP)
(1253)        TEMP = AND(STNID,:77)
(1254)        CKSUM = CKSUM+TEMP
(1255)        CALL XMTLB(TEMP)
(1256)
(1257)  C----transmit terminate op code
(1258)        CKSUM = CKSUM+TERM
(1259)        CALL XMTLB(TERM)
(1260)
(1261)  C----transmit # seconds remaining in curr. minute
(1262)
(1263)        CKSUM = CKSUM+NSEC
(1264)        CALL XMTLB(NSEC)
(1265)
(1266)  C----form and transmit action word in 2 6-bit bytes
(1267)        AWORD = AND(CWORD,:77)
(1268)        TEMP = AND(RS(AWORD,6),:77)
(1269)        CKSUM = CKSUM+TEMP
(1270)        CALL XMTLB(TEMP)
(1271)        TEMP = AND(AWORD,:77)
(1272)        CKSUM = CKSUM+TEMP
(1273)        CALL XMTLB(TEMP)
(1274)
(1275)  C----send checksum in 2 6-bit bytes
(1276)        TEMP = AND(RS(CKSUM,6),:77)
(1277)        CALL XMTLB(TEMP)
(1278)        TEMP = AND(CKSUM,:77)
(1279)        CALL XMTLB(TEMP)
(1280)
(1281)  C----and the end-of-message char.
(1282)        CALL XMTLB(EOM)
(1283)
(1284)
(1285)
```

```
(1286)  C*******************************************************
(1287)  C----wait for acknowledgement
(1288)
(1289)
(1290)  C----wait for sync code
(1291)    320 IF (MODIN(10).NE.SYNC) GOTO 320
(1292)        TEMPL = MODIN(10)
(1293)        IF (TEMPL.EQ.-1) GOTO 200
(1294)  C----check if "NAK"
(1295)  C----if not, should be a system-status packet incoming.
(1296)  C----if "NAK" attempt another transmission
(1297)        IF (TEMPL.NE.ACK) GOTO 200
(1298)        WRITE(1,10) NSEC,STNID
(1299)        WRITE (IRPRT,10) NSEC,STNID
(1300)     10   FORMAT(1X,'SYNC. TIME = ',I2,' SECS.'/
(1301)       $      1X'STATION: "',A2,'" SIGNING OFF')
(1302)
(1303)
(1304)        RETURN
(1305)
(1306)  C----abort-return
(1307)    900 WRITE (1,20) STNID
(1308)        WRITE (IRPRT,20) STNID
(1309)     20   FORMAT(1X,'FAILED TO RECEIVE TERMINATION ACKNOWLEDGEMENT'
(1310)       1    ' FROM STATION "',A2,'"')
(1311)        ICODE = 1
(1312)        RETURN
(1313)
(1314)        END
PROGRAM SIZE:   PROCEDURE - 000506    LINKAGE - 000112    STACK - 000032
0000 ERRORS [<CEASE >FTN-REV17.1]
```

```
(1315)        SUBROUTINE XLOAD(IARG,JCODE)
(1316)
(1317)
(1318)  C*****************************************************************************
(1319)  C                                                                            *
(1320)  C     ****** SUBROUTINE XLOAD ******                                         *
(1321)  C                                                                            *
(1322)  C    send a program load packet to X-TIM                                     *
(1323)  C     format of message packet as follows:                                   *
(1324)  C     variable        value(octal)                                           *
(1325)  C     sync code       SYNC    123                                            *
(1326)  C     packet size     PKSIZ   120                                            *
(1327)  C     stat. id.       STNID                                                  *
(1328)  C     op code         LPROC   143                                            *
(1329)  C         or          LPRCC   144                                            *
(1330)  C         or          LSRL    145                                            *
(1331)  C         or          LDCL    146                                            *
(1332)  C     data            WORD(1)                                                *
(1333)  C      :                :                                                   *
(1334)  C      :                :                                                   *
(1335)  C      :               WORD(=or(:176)                                       *
(1336)  C     checksum        CKSUM                                                  *
(1337)  C     end-of-mess.    EOM     103                                            *
(1338)  C                                                                            *
(1339)  C    JCODE = 1 if tel. call is to be aborted.                                *
(1340)  C          = 2 if station is to be aborted.                                  *
(1341)  C                                                                            *
(1342)  C*****************************************************************************
(1343)
(1344)
(1345)        LOGICAL ROF1,ROF2,EPF
(1346)
(1347)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(1347)        NOLIST
(1348)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(1348)        NOLIST
(1349)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(1349)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(1349)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(1349)        NOLIST
(1350)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(1350)  C Insert file for mnemonic APPLIB keys (FTN)
(1350)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(1350)        NOLIST
(1351)
(1352)
(1353)        INTEGER CHRPOS(2)
(1354)
(1355)
(1356)
(1357)
(1358)        PKSIZ = :176        /* SIZE OF MESSAGE PACKETS
(1359)        IF(AND(CWORD,:100).NE.0) PKSIZ = 40      /* Bad phone line may require
(1360)                                                 /* small packets
(1361)        NDPB = (PKSIZ-6)/2  /* # OF DATA WORDS TO BE TRANSMITTED IN MESSAGE PACKET
(1362)        JTRY = 0
```

```
(1363)
(1364)
(1365)    100 JTRY = JTRY+1
(1366)        IF (JTRY.GT.NTRY) GOTO 900
(1367)
(1368)  C----IARG = 1 : send program
(1369)  C----IARG = 2 : send DCL
(1370)  C----IARG = 3 : send SRL
(1371)        GOTO (110,120,130),IARG
(1372)
(1373)  C----setup for program load
(1374)    110 PSFIL(7) = STNID
(1375)        PRINT 10
(1376)        WRITE (IRPRT,10)
(1377)     10   FORMAT(1X,'PROGRAM LOAD STARTING')
(1378)  C----initialize load code
(1379)        ICODE = LPROG
(1380)  C----open object file
(1381)        CHRPOS(1) = 0
(1382)        CHRPOS(2) = 22
(1383)        CALL TSRC@@(K@READ,PSFIL,2,CHRPOS,ITYPE,KCODE)
(1384)        CALL ERRPR@(K@IRTN,KCODE,'BAD OPEN-PSFIL',14,'XLOAD',5)
(1385)        IF (KCODE.NE.0) GOTO 910
(1386)        GOTO 140
(1387)
(1388)  C----setup for D-C-L load
(1389)    120 DSFIL(7) = STNID
(1390)  C----initialize load code
(1391)        ICODE = LDCL
(1392)        PRINT 15
(1393)        WRITE (IRPRT,15)
(1394)     15   FORMAT(1X,'D-C-L LOAD STARTING')
(1395)        CHRPOS(1) = 0
(1396)        CHRPOS(2) = 21
(1397)        CALL TSRC@@(K@READ,DSFIL,2,CHRPOS,ITYPE,KCODE)
(1398)        CALL ERRPR@(K@IRTN,KCODE,'BAD OPEN-DSFIL',14,'XLOAD',5)
(1399)        IF (KCODE.NE.0) GOTO 910
(1400)        GOTO 140
(1401)
(1402)  C----setup for S-R-L load
(1403)    130 SSFIL(7) = STNID
(1404)  C----initialize load code
(1405)        ICODE = LSRL
(1406)        PRINT 20
(1407)        WRITE (IRPRT,20)
(1408)     20   FORMAT(1X,'S-R-L LOAD STARTING')
(1409)        CHRPOS(1) = 0
(1410)        CHRPOS(2) = 21
(1411)        CALL TSRC@@(K@READ,SSFIL,2,CHRPOS,ITYPE,KCODE)
(1412)        CALL ERRPR@(K@IRTN,KCODE,'BAD OPEN-SSFIL',14,'XLOAD',5)
(1413)        IF (KCODE.NE.0) GOTO 910
(1414)
(1415)  C----read program file into buffer
(1416)    140   READ (6,END=160,ERR=800) (WORD(I),I=1,4100)
(1417)
(1418)  C----set op code for initial packet
```

```
(1419)    160 CALL CLOSBA(2)
(1420)
(1421)        IF (.NOT.FLAG(4)) GOTO 180
(1422)        N = I-1
(1423)        DO 1000 I=1,N
(1424)   1000   CALL OCTPRT(WORD(I))
(1425)  C----set end-of-program flag as false
(1426)    180 EPF = .FALSE.
(1427)        ROF1 = .FALSE.
(1428)        ROF2 = .FALSE.
(1429)
(1430)  C*************************************************************
(1431)  C----packet transmission loop
(1432)
(1433)        DO 400 J=1,4100,NDPB
(1434)  C----define upper limit for inner loop
(1435)          JU = J+NDPB-1
(1436)          ITRY = 0
(1437)    200   ITRY = ITRY+1
(1438)          IF (ITRY.GT.NTRY) GOTO 900
(1439)
(1440)  C*************************************************************
(1441)  C----transmit program load packet
(1442)
(1443)    240   CKSUM = 0
(1444)  C----transmit sync char.
(1445)          CALL XMTLB(SYNC)
(1446)
(1447)  C----transmit packet size in 2 -  6-bit bytes,
(1448)  C----highorder first
(1449)          TEMP = RT(RS(PKSIZ,6),6)
(1450)          CKSUM = CKSUM+TEMP
(1451)          CALL XMTLB(TEMP)
(1452)          TEMP = RT(PKSIZ,6)
(1453)          CKSUM = CKSUM+TEMP
(1454)          CALL XMTLB(TEMP)
(1455)
(1456)  C----transmit station id in 2 - 6-bit bytes
(1457)          TEMP = RT(RS(STNID,8),6)
(1458)          CKSUM = CKSUM+TEMP
(1459)          CALL XMTLB(TEMP)
(1460)          TEMP = RT(STNID,6)
(1461)          CKSUM = CKSUM+TEMP
(1462)          CALL XMTLB(TEMP)
(1463)
(1464)  C----transmit program load op code
(1465)          CKSUM = CKSUM+ICODE
(1466)          CALL XMTLB(ICODE)
(1467)
(1468)  C----transmit data words in 7-bit bytes while looking for
(1469)  C----2 successive robout characters indicating
(1470)  C----the end of the program load module.
(1471)
(1472)          DO 260 I=J,JU
(1473)            TEMP = RT(RS(WORD(I),8),7)
(1474)            CKSUM = CKSUM+TEMP
```

```
(1475)            CALL XMTLB(TEMP)
(1476)            ROF1 = .FALSE.
(1477)            IF (TEMP.EQ.:177) ROF1 = .TRUE.
(1478)            IF (ROF1.AND.ROF2) EPF = .TRUE.
(1479)            TEMP = RT(WORD(I),7)
(1480)            CKSUM = CKSUM+TEMP
(1481)            CALL XMTLB(TEMP)
(1482)            ROF2 = .FALSE.
(1483)            IF (TEMP.EQ.:177) ROF2 = .TRUE.
(1484)            IF (ROF1.AND.ROF2) EPF = .TRUE.
(1485)     260    CONTINUE
(1486)
(1487)  C----send checksum in 2 6-bit bytes
(1488)        TEMP = RT(RS(CKSUM,6),6)
(1489)        CALL XMTLB(TEMP)
(1490)        TEMP = RT(CKSUM,6)
(1491)        CALL XMTLB(TEMP)
(1492)
(1493)  C----and the end-of-message char.
(1494)        CALL XMTLB(EOM)
(1495)
(1496)
(1497)
(1498)  C*********************************************************
(1499)  C----wait for ackowledgement
(1500)
(1501)
(1502)  C----wait for sync code
(1503)    320 TEMPL = MODIN(MWT)
(1504)        IF (TEMPL.EQ.-1) GOTO 200
(1505)        IF (TEMPL.NE.SYNC) GOTO 320
(1506)        TEMPL = MODIN(MWT)
(1507)        IF (TEMPL.EQ.-1) GOTO 200
(1508)  C----check if "ACK".
(1509)  C----if "NAK" attempt another transmission
(1510)  C----check if bad packet or a bad program load condition reported
(1511)          IF ((TEMPL.NE.ACK).AND.(.NOT.EPF)) GOTO 200
(1512)          IF ((TEMPL.NE.ACK).AND.EPF) GOTO 100
(1513)  C----if last packet terminates a good load, stop
(1514)          IF (EPF) GOTO 380
(1515)          ICODE = LPRGC
(1516)    400   CONTINUE
(1517)
(1518)
(1519)
(1520)    380 GOTO (420,440,460),IARG
(1521)  C----new "SRL" & "DCL" to be sent & X-TIM return-pointer to be reset
(1522)  C----& clock regs. reset
(1523)    420 CWORD = OR(CWORD,:30006)
(1524)        PRINT 25
(1525)        WRITE (IRPRT,25)
(1526)     25   FORMAT(1X,'NEW PROGRAM SENT AND LOADED')
(1527)        RETURN
(1528)    440 PRINT 30
(1529)        WRITE (IRPRT,30)
(1530)     30   FORMAT(1X,'NEW DICIAL-CONTROL-LIST SENT AND LOADED')
```

```
(1531)          RETURN
(1532)      460 PRINT 45
(1533)          WRITE (IRPRT,45)
(1534)       45   FORMAT(1X,'NEW SCAN-RATE-LIST SENT AND LOADED')
(1535)          RETURN
(1536)
(1537)
(1538)      800 PRINT 35
(1539)          WRITE (IRPRT,35)
(1540)       35   FORMAT(1X,'BAD PROGRAM-LOAD FILE')
(1541)          JCODE = 1
(1542)          RETURN
(1543)
(1544)      900 PRINT 40
(1545)          WRITE (IRPRT,40)
(1546)       40   FORMAT(1X,'UNABLE TO EXECUTE A NEW PROGRAM LOADING')
(1547)          JCODE = 1
(1548)          RETURN
(1549)
(1550)    C----must abort this station
(1551)      910 JCODE = 2
(1552)          RETURN
(1553)
(1554)          END
PROGRAM SIZE:    PROCEDURE - 001356    LINKAGE - 000156    STACK - 000036
0000 ERRORS [<XLOAD >FTN-REV17.1]
```

```
(1555)        SUBROUTINE XDUMP(IARG,JCODE)
(1556)
(1557)
(1558)
(1559)   C*************************************************************************
(1560)   C                                                                        *
(1561)   C     ****** SUBROUTINE XDUMP ******                                    *
(1562)   C                                                                        *
(1563)   C    send a message packet to X-TIM  instructing it to                  *
(1564)   C    transmit a memory dump to data develop.                           *
(1565)   C    dump will be in blocks of 64 words transmitted                    *
(1566)   C    as 2 6-bit bytes. format of message packet                         *
(1567)   C    as follows:                                                        *
(1568)   C                  variable value                                      *
(1569)   C     sync code      SYNC   123                                        *
(1570)   C     packet size    PKSIZ   10                                        *
(1571)   C     stat. id.      STNID                                             *
(1572)   C     op code        SDATA  140                                        *
(1573)   C         or         SPROC  141                                        *
(1574)   C     fwa            IFWA                                              *
(1575)   C     # words/block  BSIZE   64                                        *
(1576)   C     checksum       CKSUM                                             *
(1577)   C     end-of-mess.   EOM    103                                        *
(1578)   C                                                                      *
(1579)   C    JCODE = 1 if tp call is to be aborted.                             *
(1580)   C                                                                      *
(1581)   C*************************************************************************
(1582)
(1583)
(1584)
(1585)
(1586)   C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(1586)        NOLIST
(1587)   C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(1587)        NOLIST
(1588)   C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(1588)   C    MNEMONIC CODES FOR FILE SYSTEM (FTN)
(1588)   C    Copyright 1978, Prime Computer, Inc., Wellesley, MA
(1588)        NOLIST
(1589)   C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(1589)   C Insert file for mnemonic APPLIB keys (FTN)
(1589)   C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(1589)        NOLIST
(1590)
(1591)        INTEGER BSIZE
(1592)
(1593)
(1594)   C----initialize paRAMeter constants and open appropriate file
(1595)
(1596)        PKSIZ = 10
(1597)        JCODE = 0
(1598)        BSIZE = 64
(1599)        IF(AND(CWORD,:100).NE.0) BSIZE = 16      /* Bad phone line may require
(1600)                                                 /* small packets
(1601)
(1602)        GOTO (140,160),IARG
```

```
(1603)     140 IOP = SPROG
(1604)         PRINT 20
(1605)         WRITE (IRPRT,20)
(1606)      20    FORMAT(1X,'PROGRAM DUMP STARTING')
(1607)         NWRDS = :4000
(1608)         PDFIL(1) = STNID
(1609)         CALL OPEN@A(A@WRIT+A@SAMF,PDFIL,11,3)
(1610)         GOTO 180
(1611)     160 IOP = SDATA
(1612)         PRINT 25
(1613)         WRITE (IRPRT,25)
(1614)      25    FORMAT(1X,'DATA DUMP STARTING')
(1615)   C----WORD (5) = data block pointer, use it
(1616)         NWRDS = WORD(5)
(1617)   C----check if any data available
(1618)   C----or if pointer is arbitrary
(1619)         IF (NWRDS.EQ.:7402) NWRDS=:10000
(1620)         IF (NWRDS.EQ.0) GOTO 500
(1621)   C----if so clear data array
(1622)         DO 170 I=1,4100
(1623)           WORD(I) = 0
(1624)     170   CONTINUE
(1625)         DDFIL(1) = STNID
(1626)         CALL OPEN@A(A@WRIT+A@SAMF,DDFIL,11,3)
(1627)
(1628)   C----transmission will be in blocks of "BSIZE" words each
(1629)     180 DO 400 IBLK=1,NWRDS,BSIZE
(1630)           IFWA = (IBLK-1)     /*COMPUTE FWA OF THIS BLOCK
(1631)
(1632)   C----we shall limit our attempts to 3 for now
(1633)           ITRY = 0
(1634)     200   ITRY = ITRY+1
(1635)           IF (ITRY.GT.NTRY) GOTO 900
(1636)           CALL TIMLT(15,ERCODE)
(1637)
(1638)   C----be sure carrier is not on
(1639)     220   IF (.NOT.CD@C(0)) GOTO 240
(1640)           CALL TIMLT(0,ERCODE)
(1641)         IF (ERCODE.NE.0) GOTO 900
(1642)           GOTO 220
(1643)
(1644)   C*****************************************************
(1645)   C----transmit dump request message
(1646)
(1647)     240   CKSUM = 0
(1648)   C----transmit sync char.
(1649)           CALL XMTLB(SYNC)
(1650)
(1651)   C----transmit packet size in 2 - 6-bit bytes
(1652)   C----highorder first
(1653)           TEMP = AND(RS(PKSIZ,6),:77)
(1654)           CKSUM = CKSUM+TEMP
(1655)           CALL XMTLB(TEMP)
(1656)           TEMP = AND(PKSIZ,:77)
(1657)           CKSUM = CKSUM+TEMP
(1658)           CALL XMTLB(TEMP)
```

```
(1659)
(1660)  C----transmit station id in 2 - 6-bit bytes
(1661)          TEMP = AND(IBYSW(STNID),:77)
(1662)          CKSUM = CKSUM+TEMP
(1663)          CALL XMTLB(TEMP)
(1664)          TEMP = AND(STNID,:77)
(1665)          CKSUM = CKSUM+TEMP
(1666)          CALL XMTLB(TEMP)
(1667)
(1668)  C----transmit system status op code
(1669)          CKSUM = CKSUM+IOP
(1670)          CALL XMTLB(IOP)
(1671)
(1672)  C----send fwa of block
(1673)          TEMP = AND(RS(IFWA,6),:77)
(1674)          CKSUM = CKSUM+TEMP
(1675)          CALL XMTLB(TEMP)
(1676)          TEMP = AND(IFWA,:77)
(1677)          CKSUM = CKSUM+TEMP
(1678)          CALL XMTLB(TEMP)
(1679)
(1680)  C----send # of words to comprise a block
(1681)          TEMP = AND(RS(BSIZE,6),:77)
(1682)          CKSUM = CKSUM+TEMP
(1683)          CALL XMTLB(TEMP)
(1684)          TEMP = AND(BSIZE,:77)
(1685)          CKSUM = CKSUM+TEMP
(1686)          CALL XMTLB(TEMP)
(1687)
(1688)  C----send checksum in 2 6-bit bytes
(1689)          TEMP = AND(RS(CKSUM,6),:77)
(1690)          CALL XMTLB(TEMP)
(1691)          TEMP = AND(CKSUM,:77)
(1692)          CALL XMTLB(TEMP)
(1693)
(1694)  C----and the end-of-message char.
(1695)          CALL XMTLB(EOM)
(1696)
(1697)
(1698)
(1699)  C*********************************************************
(1700)  C----receive memory dump packet
(1701)
(1702)  C----wait for sync code
(1703)    320 TEMPL = MODIN(MWT)
(1704)        IF (TEMPL.EQ.-1) GOTO 200
(1705)        IF (TEMPL.NE.SYNC) GOTO 320
(1706)          TEMPL = MODIN(30)
(1707)        IF (TEMPL.EQ.-1) GOTO 200
(1708)  C----check if "NAK"
(1709)  C----if not, should be a system-status packet incoming.
(1710)  C----if "NAK" attempt another transmission
(1711)          IF (TEMPL.EQ.NAK) GOTO 200
(1712)
(1713)  C----should be station ID bytes -
(1714)  C---- convert from 6-bit stripped ascii to 7-bit ascii
```

```
(1715)          TEMPH = TEMPL+:300
(1716)          TEMPL = MODIN(MWT)
(1717)        IF (TEMPL.EQ.-1)GOTO 200
(1718)          ID = LS(TEMPH,8)+TEMPL+:300
(1719)
(1720)
(1721)  C----check if FWA agrees
(1722)        TEMPH = MODIN(MWT)
(1723)        IF (TEMPH.EQ.-1) GOTO 200
(1724)        TEMPL = MODIN(MWT)
(1725)        IF (TEMPL.EQ.-1) GOTO 200
(1726)        IF (IFWA.NE.(LS(TEMPH,6)+TEMPL)) GOTO 200
(1727)
(1728)  C----check if block size agrees
(1729)        TEMPH = MODIN(MWT)
(1730)        IF (TEMPH.EQ.-1) GOTO 200
(1731)        TEMPL = MODIN(MWT)
(1732)        IF (TEMPL.EQ.-1) GOTO 200
(1733)        IF (BSIZE.NE.(LS(TEMPH,6)+TEMPL)) GOTO 200
(1734)
(1735)          I = 0
(1736)    340   TEMPH = MODIN(MWT)
(1737)        IF (TEMPH.EQ.-1)GOTO 200
(1738)          IF (TEMPH.EQ.EOM) GOTO 350
(1739)          TEMPL = MODIN(MWT)
(1740)        IF (TEMPL.EQ.-1) GOTO 200
(1741)          IF (TEMPL.EQ.EOM) GOTO 350
(1742)          WORD(I+IBLK) = LS(TEMPH,6)+TEMPL
(1743)          I = I+1
(1744)          IF (I.LE.(BSIZE+2)) GOTO 340
(1745)        PRINT 62
(1746)          WRITE(IRPRT,62)
(1747)     62     FORMAT(1X,'BLOCK SIZE RECEIVED TOO LARGE')
(1748)          GOTO 200
(1749)
(1750)
(1751)  C----at this point the data array should contain:
(1752)  C---- WORD(1) = data(1)
(1753)  C----    :          :
(1754)  C----    :          :
(1755)  C---- WORD(BSIZE) = data(BSIZE)
(1756)  C---- WORD(BSIZE+1) = checksum
(1757)
(1758)  C----check for correct block size
(1759)    350   IF (I.NE.(BSIZE+1)) GOTO 200
(1760)  C----check the checksum
(1761)          IL = IBLK
(1762)          IU = IBLK+BSIZE
(1763)    352   CKSUM = AND(IBYSW(ID),:77)+AND(ID,:77)
(1764)          CKSUM = CKSUM+AND(RS(IFWA,6),:77)+AND(IFWA,:77)
(1765)          CKSUM = CKSUM+AND(RS(BSIZE,6),:77)+AND(BSIZE,:77)
(1766)          IL = IBLK
(1767)          IU = IBLK+BSIZE-1
(1768)          DO 360 I=IL,IU
(1769)            CKSUM = CKSUM+RS(WORD(I),6)+AND(WORD(I),:77)
(1770)    360     CONTINUE
```

```
(1771)  C----if a bad checksum compare, go for new transmission
(1772)    370    IF ((AND(CKSUM,:7777)).NE.WORD(IBLK+BSIZE)) GOTO 200
(1773)
(1774)  C----correct station id?
(1775)           IF (STNID.NE.ID) GOTO 200
(1776)    400    CONTINUE
(1777)
(1778)  C----word(1) may have been zeroed due to an autorestart
(1779)         IF (WORD(1).EQ.0) WORD(1)=:7777
(1780)
(1781)  C----dump file to be uniform size
(1782)         IF (IARG.EQ.2) NWRDS = :10000
(1783)
(1784)  C----put data into file
(1785)         WRITE (7) STNID,XSTAT,XTIME,ADATE(MINDX,1),
(1786)        $    OTIME(MINDX,1),(WORD(I),I=1,NWRDS)
(1787)         CALL CLOS$A(3)
(1788)
(1789)
(1790)  C----type out appropriate complete message
(1791)         GOTO (420,440),IARG
(1792)
(1793)
(1794)    420 PRINT 35
(1795)         WRITE (IRPRT,35)
(1796)     35   FORMAT(1X,'PROGRAM DUMP COMPLETED')
(1797)         RETURN
(1798)
(1799)
(1800)    440 WRITE (1,55)
(1801)         WRITE (IRPRT,55)
(1802)     55   FORMAT(1X, 'DATA DUMP COMPLETED')
(1803)
(1804)  C----save data permanently
(1805)         CALL STODAT
(1806)
(1807)  C----if call is repeated, we do not want another data dump
(1808)         CWORD = AND(CWORD,:175777)
(1809)    460    RETURN
(1810)
(1811)    500 WRITE(1,60)
(1812)         WRITE(IRPRT,60)
(1813)     60   FORMAT(1X'NO DATA AVAILABLE')
(1814)          CALL CLOS$A(3)
(1815)          RETURN
(1816)
(1817)  C----abort-return
(1818)    900 GOTO (920,940),IARG
(1819)    920 PRINT 40
(1820)         WRITE (IRPRT,40)
(1821)     40 FORMAT(1X,'UNABLE TO GET PROGRAM DUMP')
(1822)         CALL CLOS$A(3)
(1823)         JCODE = 1
(1824)         RETURN
(1825)    940 PRINT 45
(1826)         WRITE (IRPRT,45)
```

```
(1827)      45   FORMAT(1X,'UNABLE TO GET DATA DUMP')
(1828)           CALL CLOS$A(3)
(1829)           JCODE = 1
(1830)           RETURN
(1831)
(1832)           END
PROGRAM SIZE:    PROCEDURE - 001454    LINKAGE - 009166    STACK - 000036
0000 ERRORS [<XDUMP >FTN-REV17.1]
```

```
(1833)        SUBROUTINE STODAT
(1834)
(1835)  C*****************************************************************************
(1836)  C                                                                            *
(1837)  C     Store data dump in .DATA. subufd of cross under filename               *
(1838)  C     of format IDYY.DDD.HH and add this name to the directory               *
(1839)  C     file "DIRECT".                                                         *
(1840)  C                                                                            *
(1841)  C*****************************************************************************
(1842)
(1843)
(1844)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(1844)        NOLIST
(1845)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(1845)        NOLIST
(1846)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(1846)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(1846)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(1846)        NOLIST
(1847)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(1847)  C Insert file for mnemonic APPLIB keys (FTN)
(1847)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(1847)        NOLIST
(1848)
(1849)
(1850)        INTEGER*2 TRENAM(14), HOUR(1), ARRAY(3), CHRPOS(2)
(1851)        INTEGER*2 ITEXT(384),JTEXT(376)
(1852)        INTEGER*2 PERIOD,SPACE
(1853)        INTEGER*2 TXTNAM(13),DIRNAM(11)
(1854)
(1855)        LOGICAL SW
(1856)
(1857)        EQUIVALENCE (ITEXT(9),JTEXT(1))
(1858)
(1859)
(1860)
(1861)        DATA TRENAM/'<*>CROSS>.DATA.>              '/
(1862)        DATA TXTNAM/'<*>CROSS>.TEXT.>   .DESCRP '/
(1863)        DATA DIRNAM/'<*>CROSS>.DATA.>DIRECT'/
(1864)        DATA PERIOD,SPACE/'.',' '/
(1865)
(1866)  C----open file containing descriptive text for current station and read in.
(1867)        CALL FILL$A(ITEXT,768,SPACE)
(1868)        TXTNAM(9) = STNID
(1869)        CHRPOS(1) = 0
(1870)        CHRPOS(2) = 25
(1871)        CALL TSRC$$(K$READ,TXTNAM,7,CHRPOS,ITYPE,ICODE)
(1872)        CALL ERRPR$(K$IRTN,ICODE,'NO TEXT',7,STNID,2)
(1873)        IF (ICODE.NE.0) GOTO 220
(1874)        CALL PRWF$$(K$READ+K$PREA+K$CONV,7,LOC(JTEXT),
(1875)       $     376,INTL(0),NMW,ICODE)
(1876)        CALL ERRPR$(K$IRTN,ICODE,'BAD TEXT READ',13,'STODAT',6)
(1877)        IF (ICODE.NE.0) CALL CLSOUT
(1878)    200 CHRPOS(1) = 0
(1879)        CALL TSRC$$(K$CLOS,TXTNAM,7,CHRPOS,ITYPE,ICODE)
(1880)
```

```
(1881)
(1882)    220 ITIME = OTIME(MINDX,1)
(1883)
(1884)  C----generate treename of the form "CROSS>.DATA.>IDYY.DDD.HH"
(1885)
(1886)        ENCODE(6,1,ARRAY) ADATE(MINDX,1)
(1887)      1   FORMAT(F6.3)
(1888)        TRENAM(9) = STNID
(1889)        TRENAM(10) = ARRAY(1)
(1890)        TRENAM(11) = ARRAY(2)
(1891)        TRENAM(12) = ARRAY(3)
(1892)
(1893)        ENCODE(2,2,HOUR) ITIME
(1894)      2   FORMAT(B'##')
(1895)        HOUR(1) = IBYSW(HOUR(1))
(1896)        TRENAM(13) = OR(AND(HOUR(1),:377),AND(PERIOD,:177400))
(1897)        TRENAM(14) = OR(AND(HOUR(1),:177400),AND(SPACE,:377))
(1898)        SW = .FALSE.
(1899)
(1900)  C----open the file
(1901)    225 CHRPOS(1) = 0
(1902)        CHRPOS(2) = 28
(1903)        CALL TSRC##(K#WRIT,TRENAM,7,CHRPOS,ITYPE,ICODE)
(1904)        CALL ERRPR#(K#IRTN,ICODE,TRENAM,28,'STODAT',6)
(1905)        IF (ICODE.EQ.0) GOTO 230
(1906)
(1907)  C----no origin time, synthesize a name if not previously tried.
(1908)        IF (SW) CALL CLSOUT
(1909)        TRENAM(10) = '.D'
(1910)        TRENAM(11) = 'AT'
(1911)        TRENAM(12) = 'A.'
(1912)        TRENAM(13) = 'TE'
(1913)        TRENAM(14) = 'MP'
(1914)  C----tell about the new name
(1915)        WRITE(1,29) (TRENAM(I),I=9,14)
(1916)        WRITE(IRPRT,20) (TRENAM(I),I=9,14)
(1917)     20   FORMAT(1X,'SUBSTITUTE NAME USED: ',6A2)
(1918)        SW = .TRUE.
(1919)        GOTO 225
(1920)
(1921)  C----convert time and date to integer form
(1922)    230 IYEAR = ADATE(MINDX,1)
(1923)        IDAY = (ADATE(MINDX,1)-FLOAT(IYEAR))*1000.+0.5
(1924)        ISECH = OTIME(MINDX,1)*3600./2.**15
(1925)        ATIME = OTIME(MINDX,1)*3600.
(1926)        ISECL = AMOD(ATIME,2.**15)
(1927)        JSECH = XTIME/2.**15
(1928)        JSECL = AMOD(XTIME,2.**15)
(1929)
(1930)        ITEXT(1) = STNID
(1931)        ITEXT(2) = IYEAR
(1932)        ITEXT(3) = IMON
(1933)        ITEXT(4) = IDAY
(1934)        ITEXT(5) = ISECH
(1935)        ITEXT(6) = ISECL
(1936)        ITEXT(7) = JSECH
```

```
(1937)          ITEXT(8) = JSECL
(1938)
(1939)
(1940)    C----put "ITEXT" & "WORD" arrays into output file
(1941)          CALL PRWF@@(K@WRIT+K@PREA,7,LOC(ITEXT),384,INTL(0),IRNW,KODE)
(1942)          IF (KODE.NE.0) GOTO 240
(1943)          CALL PRWF@@(K@WRIT+K@PRER,7,LOC(WORD),4096,INTL(0),IRNW,KODE)
(1944)      240  CALL ERRPR@(K@IRTN,KODE,'BAD-WRITE',9,'STODAT',6)
(1945)          IF (ICODE.NE.0) CALL CLSOUT
(1946)
(1947)    C----close output data file
(1948)          CHRPOS(1) = 0
(1949)          CALL TSRC@@(K@CLOS,TRENAM,7,CHRPOS,ITYPE,ICODE)
(1950)          CALL ERRPR@(K@IRTN,ICODE,TRENAM,28,'STODAT',6)
(1951)          IF (ICODE.NE.0) CALL CLSOUT
(1952)
(1953)    C----add data file name to directory
(1954)          CHRPOS(1) = 0
(1955)          CHRPOS(2) = 22
(1956)          CALL TSRC@@(K@RDWR,DIRNAM,7,CHRPOS,ITYPE,ICODE)
(1957)          CALL ERRPR@(K@IRTN,ICODE,'DIRECT FILE',11,'STODAT',6)
(1958)          IF (ICODE.NE.0) CALL CLSOUT
(1959)          IF (GEND@A(7)) GOTO 300
(1960)          CALL TNOU('COULD NOT LOCATE END-OF-FILE FOR "DIRECT"',41)
(1961)          CALL CLSOUT
(1962)
(1963)      300 WRITE(11,10)(TRENAM(I),I=9,14)
(1964)       10   FORMAT(6A2)
(1965)          IF (TRNC@A(7)) GOTO 320
(1966)          CALL TNOU('COULD NOT TRUNCATE "DIRECT"',27)
(1967)          CALL CLSOUT
(1968)
(1969)      320 CHRPOS(1) = 0
(1970)          CALL TSRC@@(K@CLOS,DIRNAM,7,CHRPOS,ITYPE,ICODE)
(1971)          CALL ERRPR@(K@IRTN,ICODE,'BAD CLOSE-DIRECT',16,'STODAT',6)
(1972)          IF (ICODE.NE.0) CALL CLSOUT
(1973)
(1974)          RETURN
(1975)          END
PROGRAM SIZE:    PROCEDURE - 001336     LINKAGE - 001034     STACK - 000032
0000 ERRORS [<STODAT>FTN-REV17.1]
```

```
(1976)         SUBROUTINE OCTAL(M,MARRAY)
(1977)   C
(1978)   C  * convert M to a four digit octal number stored in A1 format
(1979)   C  * in MARRAY
(1980)   C ** this procedure is necessary because standard FORTRAN
(1981)   C ** does not have an 'O'. (octal) format.  if the compiler
(1982)   C ** used does have an 'O' format, use of this format
(1983)   C ** should replace the use of this routine.  e.g.
(1984)   C ** replace
(1985)   C **         CALL OCTAL(PC,OUTPC)
(1986)   C **         WRITE (OUTDEV,4000) (OUTPC(I), I=1,4)
(1987)   C **  4000 FORMAT (1X, 4A1)
(1988)   C ** by
(1989)   C **         WRITE (OUTDEV,4000) PC
(1990)   C **  4000 FORMAT (1X, O4)
(1991)   C
(1992)         INTEGER MARRAY(6)
(1993)         INTEGER TEMP
(1994)   C
(1995)         I1=M
(1996)         DO    I=1,6
(1997)         J=7
(1998)         TEMP=MOD(I1,8)
(1999)         MARRAY(J)='0'+LS(TEMP,8)
(2000)         I1=I1/8
(2001)      10 CONTINUE
(2002)         RETURN
(2003)         END
PROGRAM SIZE:   PROCEDURE - 000041    LINKAGE - 000030    STACK - 000032
0000 ERRORS [<OCTAL >FTN-REV17.1]
```

```
(2004)         INTEGER FUNCTION OCTDCD(CHAR)
(2005)    C*****************************************************************************
(2006)    C                                                                            *
(2007)    C                        ****** OCTDCD ******                               *
(2008)    C                                                                            *
(2009)    C     Function to convert 5 character array to octal equivalent              *
(2010)    C                                                                            *
(2011)    C*****************************************************************************
(2012)
(2013)
(2014)
(2015)         INTEGER CHAR(5)
(2016)
(2017)         NUM = 0
(2018)         DO 200 I=1,5
(2019)           NUM = LS(NUM,3)+RS((CHAR(I)-'0'),8)
(2020)     200   CONTINUE
(2021)         OCTDCD = NUM
(2022)         RETURN
(2023)         END
PROGRAM SIZE:    PROCEDURE - 000031    LINKAGE - 000026    STACK - 000032
0000 ERRORS [<OCTDCD>FTN-REV17.1]
```

```
(2024)         FUNCTION IBYSW(IARG)
(2025)   C*****************************************************************
(2026)   C
(2027)   C                    ****** IBYSW ******
(2028)   C
(2029)   C     Byte swap routine
(2030)   C
(2031)   C*****************************************************************
(2032)
(2033)
(2034)
(2035)          IBYSW = AND(LS(IARG,8),:177400)+AND(RS(IARG,8),:377)
(2036)          RETURN
(2037)          END
PROGRAM SIZE:    PROCEDURE - 000015     LINKAGE - 000022     STACK - 000032
0000 ERRORS [<IBYSW >FTN-REV17.1]
```

```
(2038)  C********************************************************************************
(2039)  C----modem control package of subroutines                                      *
(2040)  C----uses AMLC line 34 to control modem                                        *
(2041)  C----uses AMLC inline 35 to communicate                                        *
(2042)  C----DTR & RTS may be set by program                                           *
(2043)  C----DTR,RTS,CTS,DSR, & CD may be tested                                       *
(2044)  C----a change in state of CTS,DSR, or CD will force                            *
(2045)  C------an output.  program may acquire modem status by                        *
(2046)  C------sending a "XMT".                                                       *
(2047)  C----last acquired status is kept in location "MSTAT"                          *
(2048)  C********************************************************************************
```

```
(2050)  C                       ****** DTR@C ******
(2051)
(2052)  C----if arg is true, DTR is turned on, if false, dtr is turned off
(2053)
(2054)        SUBROUTINE DTR@C(ARG)
(2055)
(2056)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(2056)        NOLIST
(2057)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(2057)        NOLIST
(2058)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(2058)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(2058)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(2058)        NOLIST
(2059)  C A@KEYS, APPLIB>SOURCE, ELS, 02/12/79
(2059)  C Insert file for mnemonic APPLIB keys (FTN)
(2059)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(2059)        NOLIST
(2060)
(2061)        INTEGER DTR,CMND,STAT(2)
(2062)        LOGICAL ARG
(2063)
(2064)        COMMON /MSTAT/MDSTAT
(2065)
(2066)        DATA DTR/:40000/
(2067)
(2068)        MDSTAT = AND(MDSTAT,:37777)
(2069)        IF (ARG) MDSTAT = OR(MDSTAT,DTR)
(2070)        CALL T@AMLC(CTLLIN,LOC(MDSTAT),1,3,STAT)
(2071)
(2072)        RETURN
(2073)        END
PROGRAM SIZE:   PROCEDURE - 000040    LINKAGE - 000034    STACK - 000032
0000 ERRORS [<DTR@C >FTN-REV17.1]
```

```
(2074)  C                       ****** RTS$C ******
(2075)
(2076)  C----if arg is true, RTS is turned on, if false, RTS is turned off
(2077)
(2078)        SUBROUTINE RTS$C(ARG)
(2079)
(2080)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(2080)        NOLIST
(2081)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(2081)        NOLIST
(2082)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(2082)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(2082)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(2082)        NOLIST
(2083)  C A$KEYS, APPLIB>SOURCE, ELS, 02/12/79
(2083)  C Insert file for mnemonic APPLIB keys (FTN)
(2083)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(2083)        NOLIST
(2084)
(2085)        INTEGER RTS,CMND,STAT(2)
(2086)        LOGICAL ARG
(2087)
(2088)        COMMON /MSTAT/MDSTAT
(2089)
(2090)        DATA RTS/:20000/
(2091)
(2092)        MDSTAT = AND(MDSTAT,:57777)
(2093)        IF (ARG) MDSTAT = OR(MDSTAT,RTS)
(2094)        CALL TRAMLC(CTLLIN,LOC(MDSTAT),1,3,STAT)
(2095)
(2096)        RETURN
(2097)        END
PROGRAM SIZE:   PROCEDURE - 000040    LINKAGE - 000034    STACK - 000032
0000 ERRORS [<RTS$C >FTN-REV17.1]
```

```
(2098)  C                    ****** MDMS@C ******
(2099)
(2100)  C----returns true if function is active otherwise false
(2101)
(2102)
(2103)        LOGICAL FUNCTION MDMS@C(ARG)
(2104)
(2105)  C X-TIM.TALK.DECLAR          /* common variable declaration for X-TIM.TALK
(2105)        NOLIST
(2106)  C SYSCOM>KEYS.F       MNEMONIC KEYS FOR FILE SYSTEM (FTN)      09/29/78
(2106)        NOLIST
(2107)  C ERRD.F, SYSCOM, OS GROUP, 03/29/79
(2107)  C     MNEMONIC CODES FOR FILE SYSTEM (FTN)
(2107)  C     Copyright 1978, Prime Computer, Inc., Wellesley, MA
(2107)        NOLIST
(2108)  C A@KEYS, APPLIB>SOURCE, ELS, 02/12/79
(2108)  C Insert file for mnemonic APPLIB keyw (FTN)
(2108)  C Copyright 1977, PRIME COMPUTER, INC., Framingham, MA.
(2108)        NOLIST
(2109)
(2110)        INTEGER XMT,STAT(2),CMND,ARG
(2111)
(2112)        COMMON /MSTAT/MDSTAT
(2113)
(2114)        DATA XMT/:400/
(2115)
(2116)
(2117)  C----be sure input buffer is empty
(2118)        CALL MTBF@C(CTLLIN)
(2119)  C----request modem status
(2120)        CMND = OR(MDSTAT,XMT)
(2121)        CALL T@AMLC(CTLLIN,LOC(CMND),1,3,STAT)
(2122)  C----wait for it to be available
(2123)    200 CALL T@AMLC(CTLLIN,LOC(MDSTAT),1,4,STAT)
(2124)        IF (STAT(1).NE.0) GOTO 220
(2125)        CALL SLEEP@(00000030)
(2126)        GOTO 200
(2127)
(2128)  C----fetch modem status word
(2129)    220 CALL T@AMLC(CTLLIN,LOC(MDSTAT),1,1,STAT)
(2130)  C----check if function bit is on and return appropriately
(2131)        MDMS@C = .FALSE.
(2132)        IF (AND(MDSTAT,ARG).NE.0) MDMS@C = .TRUE.
(2133)        RETURN
(2134)        END
PROGRAM SIZE:   PROCEDURE - 000120    LINKAGE - 000042    STACK - 000032
0000 ERRORS [<MDMS@C>FTN-REV17.1]
```

```
(2135)  C                    ****** DSR@C ******
(2136)
(2137)
(2138)  C----logical function routine
(2139)  C----returns true if DSR is active otherwise false
(2140)
(2141)
(2142)          LOGICAL FUNCTION DSR@C(IARG)
(2143)
(2144)          INTEGER DSR
(2145)
(2146)          LOGICAL MDMS@C
(2147)
(2148)          COMMON /MSTAT/MDSTAT
(2149)
(2150)          DATA DSR/:10000/
(2151)
(2152)
(2153)          DSR@C = .FALSE.
(2154)          IF (MDMS@C(DSR)) DSR@C =     UE.
(2155)          RETURN
(2156)          END
PROGRAM SIZE:    PROCEDURE - 000015     LINKAGE - 000024     STACK - 000030
0000 ERRORS [<DSR@C >FTN-REV17.1]
```

```
(2157)  C                   ****** CTS@C ******
(2158)
(2159)
(2160)  C----logical function routine
(2161)  C----returns true if CTS is active otherwise false
(2162)
(2163)
(2164)          LOGICAL FUNCTION CTS@C(IARG)
(2165)
(2166)          INTEGER CTS
(2167)
(2168)          LOGICAL MDMS@C
(2169)
(2170)          COMMON /MSTAT/MDSTAT
(2171)
(2172)          DATA CTS/:4000/
(2173)
(2174)
(2175)          CTS@C = .FALSE.
(2176)          IF (MDMS@C(CTS)) CTS@C = .TRUE.
(2177)          RETURN
(2178)          END
PROGRAM SIZE:    PROCEDURE - 000015    LINKAGE - 000024    STACK - 000030
0000 ERRORS [<CTS@C >FTN-REV17.1]
```

```
(2179)  C                    ****** CD@C ******
(2180)
(2181)
(2182)  C----logical function routine
(2183)  C----returns true if CD is active otherwise false
(2184)
(2185)
(2186)          LOGICAL FUNCTION CD@C(IARG)
(2187)
(2188)          INTEGER CD
(2189)
(2190)          LOGICAL MDMS@C
(2191)
(2192)          COMMON /MSTAT/MDSTAT
(2193)
(2194)          DATA CD/:2000/
(2195)
(2196)
(2197)          CD@C = .FALSE.
(2198)          IF (MDMS@C(CD)) CD@C = .TRUE.
(2199)          RETURN
(2200)          END
PROGRAM SIZE:   PROCEDURE - 000015    LINKAGE - 000024    STACK - 000030
0000 ERRORS [<CD@C  >FTN-REV17.1]
```

```
(2201)  C                   ****** MTBF@C ******
(2202)
(2203)
(2204)  C----empty the almc input buffer
(2205)
(2206)
(2207)        SUBROUTINE MTBF@C(N)
(2208)
(2209)        INTEGER JUNK,STAT(2)
(2210)
(2211)    200 CALL T@AMLC(N,LOC(JUNK),1,4,STAT)
(2212)        IF (STAT(1).EQ.0) RETURN
(2213)        CALL T@AMLC(N,LOC(JUNK),1,1,STAT)
(2214)        GOTO 200
(2215)
(2216)        END
PROGRAM SIZE:   PROCEDURE - 000042    LINKAGE - 000030    STACK - 000032
0000 ERRORS [<MTBF@C>FTN-REV17.1]
```

```
(2217)  C********************************************************************************
(2218)  C                                                                               *
(2219)  C                        ****** OCTPRT ******                                   *
(2220)  C                                                                               *
(2221)  C       Prints argument in octal                                                *
(2222)  C                                                                               *
(2223)  C********************************************************************************
(2224)
(2225)
(2226)        SUBROUTINE OCTPRT(IARG)
(2227)
(2228)        DIMENSION ICHAR(6)
(2229)
(2230)        CALL OCTAL(IARG,ICHAR)
(2231)        PRINT 10,ICHAR
(2232)     10   FORMAT(6A1)
(2233)        RETURN
(2234)        END
PROGRAM SIZE:   PROCEDURE - 000032    LINKAGE - 000040    STACK - 000030
0000 ERRORS [<OCTPRT>FTN-REV17.1]
```

TUE, APR 01 1980

```
                ////////////////////////////////////////////////////////////////////
                /
                /            ############################
                /            #######X-TIM.COMM*PA#######
                /            ############################
                /
                /       Written in PDP-8 assembly language (PAL3) for execution
                /       by a IM6100 microprocessor TIM system.
                /
                /       Author: Robert F. Nickerson
                /               CALTECH Seismological Laboratory
                /               Pasadena, California 91125
                /
                /               October 31, 1979
                /
                /


0000            ID = 0000       /station identifier ("--")
0010            VRSION =10      /version #
                /     TIM data logging program
                /
                ////////////////////////////////////////////////////////////////////
                /
                /    "/***" indicates a feature out of service on this unit
                /
                ////////////////////////////////////////////////////////////////////
                /
                /       switch options:
                /           CTR on, SW(8) on
                /               reset: fill program memory with halts, display station id.
                /           CTR on, SW(8) off
                /               reset: restart at address 200+C[SW1-4]
                /
                ////////////////////////////////////////////////////////////////////
```

```
37                      PAUSE
38
39
40              ////////////////////////////////////////////////////////
41              /
42              /PROM resident communications sequencer
43              /handles communications via UART and MODEM
44              /
45              /PIE usage
46              /
47              / PIE READ1 READ2 WRITE1 WRITE2  S1  S2  S3  S4  F1  F2  F3  F4
48              / PMOD --    --     --     --    CTS DSR CD ring RTS --  DTR --
49              /PUART DRR   --     TBRL   --    DR TBRE TRE err MR  --  --  --
50              /PSTAT read  --    clear  display --  --  --  --  --  --  --  --
51              /
52              /
53              /  message packet formats (all single byte words = 100+byte, e.g. EOM=103)
54              /
55              //Data Central to X-TIM
56              /
57              /  request for system status
58              /    # bytes in packet (=or<128)
59              /    station id #
60              /    op code (142)
61              /    checksum
62              /    EOM
63              /
64              /  request for program or data dump
65              /    sync code
66              /    # bytes in packet
67              /    station id
68              /    op code (140 or 141)
69              /    fwa of block to be sent
70              /    # words to be sent
71              /    checksum
72              /    EOM
73              /
74              /  transmit program or table
75              /    sync code
76              /    # bytes in packet
77              /    station id
78              /    op code (143-146)
79              /    data
80              /     :
81              /     :
82              /    checksum
83              /    EOM
```

```
                    PAUSE
                /
                /  terminate
                /    sync code
                /    # bytes in packet
                /    station id
                /    op code (147)
                /    # secs. in curr. min.
                /    action word
                /    checksum
                /    EOM
                /
                /     1/2 word transmission order - hi,  lo
                /
                //system status to data central
                /  sync code
                /  station id #
                /  time - hi order
                /  time - lo order
                /  system status word
                /  data block pointer
                /  checksum
                /  EOM
                /
                /
                //data packets to data central
                /  sync code
                /  station id #
                /  fwa of block
                /  # words in block
                /  data
                /  checksum
                /  EOM
                /
                ////////////////////////////////////////////////////////
```

```
119                         PAUSE
120
121
122
123          7000   ORIGIN= 7000
124
125                 /************************************************************
126                 /
127                 /first we establish the reason for the reset-restart by
128                 /looking at the status register. if a manual reset and
129                 /if "CTR" is active and SW8 is on,   then program area is filled
130                 /with halts and system is suspended.
131                 /if SW8 is off,   a program branch is taken to
132                 /200+C[SW]. if telephone,   it is serviced. if an autoreset,
133                 /system is reset and all activity is suspended.
134                 /
135                 /************************************************************
136
137          7000   *ORIGIN
138    7000  7300   START,  CLA CLL          /reset entry point
139    7001  6240           STIN             /fetch status word
140    7002  0241           AND     K7277    /clean out non-hardware bits
141    7003  7421           MQL              /put current status in mq
142    7004  1177           TAD     SVSTAT   /fetch existing status word
143    7005  7501           MQA              /or with new status word
144    7006  3177           DCA     SVSTAT   /save updated status
145    7007  6241           STCLR            /clear status reg.
146    7010  7501           MQA              /fetch current status word
147    7011  7510           SPA              /manual reset?
148    7012  5221           JMP     RST      /yes - go check if branch
149    7013  7006           RTL              /no
150    7014  7710           SPA CLA          /tp ring?
151    7015  5244           JMP     TPRING   /yes
152    7016  7430           SZL              /no - autoreset?
153    7017  5643           JMP I   SUSP     /yes - suspend all
154    7020  5407           JMP I   RETPTR   /return to ram
155
156    7021  0237   RST,    AND     K20      /get CTR bit
157    7022  7650           SNA CLA          /CTR switch on?
158    7023  5643           JMP I   SUSP     /no - suspend all
159    7024  7501           MQA              /yes
160    7025  0236           AND     K10      /get SW8 bit
161    7026  7640           SZA CLA          /active
162    7027  5642           JMP I   FLHLTP   /yes - fill with halts
163    7030  7501           MQA              /no - fetch status reg.
164    7031  0313           AND     K7       /get switch bits
165    7032  1240           TAD     K200     /convert to address
166    7033  3007           DCA     RETPTR
167    7034  6421           MOFF             /power down MODEM
168    7035  5407           JMP I   RETPTR   /go there
169
170    7036  0010   K10,    10
171    7037  0020   K20,    20
172    7040  0200   K200,   200
173    7041  7277   K7277,  7277
174    7042  7577   FLHLTP, FLHLTS
175    7043  7614   SUSP,   SUSPND
```

```
176                             PAUSE
177
178
179                     /     ****** X-TIM.COMM*PA ******
180                     /
181                     /communications controller sequence
182
183                     /check if auto- or manual- restart occurred,   if so
184                     /defeat entry to datalogger
185
186   7044  7333      TPRING, GEN6K
187   7045  0177              AND     SVSTAT
188   7046  7650              SNA CLA
189   7047  5252              JMP     TPRI
190   7050  1275              TAD     KINXMT
191   7051  3006              DCA     DLPTR
192
193                     /move subroutine pointers to page 0
194   7052  1276      TPRI,   TAD     PTRCNT   /fetch list size
195   7053  3155              DCA     CNTRI    /init. counter
196   7054  1277              TAD     SOURCE   /fetch source addr.
197   7055  3017              DCA     17
198   7056  1300              TAD     DESTIN   /init. destination pointer
199   7057  3016              DCA     16
200
201                     /do the transfer
202   7060  1417              TAD I   17
203   7061  3416              DCA I   16
204   7062  2155              ISZ     CNTRI
205   7063  5260              JMP     .-3
```

```
206                         PAUSE
207
208                 /initialize the neccessary PIE's (PMOD,   PUART)
209                 /control reg. B as follows:
210                 /
211                 /       PMOD    PUART
212                 /       SP1       1       1
213                 /       SP2       1       1
214                 /       SP3       1       1
215                 /       SP4       1       1
216                 /       SL1       1       0
217                 /       SL2       1       0
218                 /       SL3       1       0
219                 /       SL4       0       0
220
221                 /initialize MODEM and UART PIEs
222   7064  1302            TAD     KSPALL    /init. for pos. edge sense
223   7065  6455            WCRB PUART        /UART PIE
224   7066  1301            TAD     KSL123    /also sense 1,  2,3 to be pos. level sensitive
225   7067  6435            WCRB PMOD         /MODEM PIE
226   7070  7200            CLA
227   7071  6436            SDTR              /set "DTR" to answer call
228   7072  6423            DSRSF             /wait for "DSR"
229   7073  5272            JMP     .-1
230   7074  5337            JMP     MAIN2
231
232   7075  7344    KINXMT, INXMTP            /pointer to xmtr init. routine
233   7076  7725    PTRCNT, -SIZPC0           /size of subroutine pointer list
234   7077  7523    SOURCE, STPCE0-1          /fwa of list
235   7100  0076    DESTIN, FWAPC0-1
236   7101  3400    KSL123, SL1 SL2 SL3
237   7102  0360    KSPALL, SP1 SP2 SP3 SP4
```

```
238                             PAUSE
239
240                     /initialize UART to receive message then wait for sync
241                     /code. returns only if received.
242     7103    1324    INRECP, TAD     KD30
243     7104    4102            JMS     TIMER
244     7105    5545            JMP I TOP
245     7106    6432            CDSF                    /wait for carrier to come on
246     7107    5304            JMP     .-3             /  but only for 30 secs.
247     7110    6446            SFL1 PUART              /reset UART
248     7111    6447            CFL1 PUART
249     7112    6442            SKIP1 PUART             /reset sense flop - maybe?
250     7113    0007    K7,     7                       /nop
251     7114    4102            JMS     TIMER           /we don't wait forever
252     7115    5545            JMP I TOP
253     7116    6442            SKIP1 PUART             /wait for null char.
254     7117    5314            JMP     .-3
255     7120    6440            READ1 PUART             /reset read buffer
256     7121    6446            SFL1 PUART              /  and error lines
257     7122    6447            CFL1 PUART
258     7123    6453            SKIP4 PUART             /  and PE sense flop
259     7124    0036    KD30,   36                      /NOP
260                     /we now wait for a sync char. to come in
261     7125    7305            CEN2                    /time-out to be init.
262     7126    4135            JMS     GTCHAR          /fetch incoming character
263     7127    7041            CIA                     /check if sync char.
264     7130    1144            TAD     KSYNC
265     7131    7640            SZA CLA
266     7132    5326            JMP     .-4             /not - keep on waiting
267     7133    5532            JMP I   INREC          /arrived,   now return
268
269
270
271
272
273
274
275
276
277
278
279                     /bad reception - send "NAK"
280
281     7134    7240    BADREC, CLA CMA
282     7135    4077            JMS     REPLY
283     7136    5337            JMP     MAIN2
```

```
                        PAUSE


                /sequence to collect message packet and store in buffer


MAIN2,  TAD     BUFRST  /init. pointer to buffer
        DCA     17
        DCA     CHKSUM  /clear checksum
        JMS     INREC
/next 2 bytes will be the byte count of the incoming packet
        JMS     GTCHAR  /fetch next char. (hi order byte count)
        STL
        JMS     UPCKSM  /update checksum
        BSW             /put into hi order bits
        DCA     SAVE1   /store
        JMS     GTCHAR  /fetch lo order byte
        JMS     UPCKSM  /update checksum
        TAD     SAVE1   /blend them
        DCA     PKTSIZ  /save
        TAD     PKTSIZ
        CIA
        DCA     CNTR2   /also become counter
/now we start stuffing the buffer with the incoming
/packet of bytes.
MN1,    CDN2
        JMS     GTCHAR  /start stuffing the buffer
        SPA             /check for parity error
        JMP      BADREC /break out if so
        STL
        JMS     UPCKSM
        DCA I   17
        ISZ     CNTR2   /more?
        JMP     MN1     /yes


///checkquality of buffer contents
/   was buffer exceeded?
        TAD     17
        SPA CLA         /test for < 4000
        JMP I   BDRECP  /oops!
/     is EOM in correct location?
        TAD      BUFRST
        TAD      PKTSIZ
        DCA      SAVE1
        TAD I    SAVE1
        CIA
        TAD      KEOM
        SZA CLA
        JMP I   BDRECP
```

```
335                     PAUSE
336
337
338
339             /checksum valid?
340   7203  7346  CONT,   CENM3
341   7204  1166          TAD     SAVE1
342   7205  3016          DCA     16
343   7206  1016          TAD     16
344   7207  3015          DCA     15
345   7210  1416          TAD I   16      /reconstruct the transmitted checksum
346   7211  7002          BSW
347   7212  1416          TAD I   16
348   7213  7041          CIA
349   7214  1162          TAD     CHKSUM  /subtract from the computed checksum
350   7215  7041          CIA             /subtract transmitted checksum component
351   7216  1415          TAD I   15
352   7217  1415          TAD I   15
353   7220  1415          TAD I   15      /and EOM
354   7221  7640          SZA CLA         /compare
355   7222  5551          JMP I   BDRECP
356
357             /check station id # - top entry in buffer
358   7223  1147          TAD     BUFRST
359   7224  3017          DCA     17
360   7225  1417          TAD I   17      /fetch id from buffer
361   7226  7002          BSW
362   7227  1417          TAD I   17
363   7230  7041          CIA
364   7231  1543          TAD I IDP       /fetch real #
365   7232  7640          SZA CLA
366   7233  5664          JMP I   PTRLST+3 /not equal - answer with system status only
367
```

```
                        PAUSE

                /all is well - fetch op code,   check its validity and process
7234  1417      TAD I   17
7235  3166      DCA     SAVE1
7236  1166      TAD     SAVE1
7237  0273      AND     K7760
7240  7041      CIA
7241  1365      TAD     K140
7242  7640      SZA CLA
7243  5551      JMP I   BDRECP
7244  1166      TAD     SAVE1
7245  0335      AND     K17
7246  3166      DCA     SAVE1     /check that op-code does not exceed list
7247  1272      TAD     ENDLST
7250  1166      TAD     SAVE1
7251  7700      SMA CLA
7252  5551      JMP I   BDRECP
7253  1166      TAD     SAVE1
7254  1261      TAD     PTRLST
7255  3166      DCA     SAVE1
7256  1566      TAD I   SAVE1
7257  3166      DCA     SAVE1
7260  5566      JMP I   SAVE1

                //op code pointer list
7261  7262      PTRLST, .+1
7262  7275              SDDATA          /140 - transmit data memory
7263  7274              SDPROG          /141 - transmit program memory
7264  7650              SDSTAT          /142 - transmit system status packet
7265  7675              LDPROG          /143 - load program
7266  7702              LDPRCC          /144 - continuation of program load
7267  7674              LDSRL           /145 - load sample rate list
7270  7672              LDDCL           /146 - load digital control list
7271  7371              TERM            /147 - terminate
7272  7767      ENDLST, PTRLST-.


                /constants
7273  7760      K7760,  7760
```

```
409                         PAUSE
410
411
412
413                 /transmit data storage memory or program memory
414                 /as specified by message from data central.
415                 /
416                 /packet format is:
417                 /       sync code
418                 /       station id #
419                 /       fwa of block
420                 /       # of data words in block
421                 /       data(1)
422                 /         :
423                 /         :
424                 /       data(n)
425                 /       checksum
426                 /       EOM
427
428  7274  1342   SDPROG, TAD     SD5P      /entry for prog. mem. xmt
429  7275  1343   SDDATA, TAD     SD6P      /entry for data mem. xmt
430  7276  3167           DCA     SAVE2
431  7277  1417   SD1,    TAD I   17        /fetch and assemble fwa of block
432  7300  7002           BSW
433  7301  1417           TAD I   17
434  7302  3161           DCA     DATPTR    /use as pointer
435  7303  1417           TAD I   17        /fetch and assemble # of words requested
436  7304  7002           BSW
437  7305  1417           TAD I   17
438  7306  7041           CIA
439  7307  3156           DCA     CNTR2     /use neg. as a counter
440
441                 /transmit header characters
442  7310  4121           JMS     INXMT
443  7311  1543           TAD I   IDP       /transmit station id
444  7312  4113           JMS     UPCKSM
445  7313  4116           JMS     XMT12
446  7314  1161           TAD     DATPTR    /transmit fwa of block
447  7315  4113           JMS     UPCKSM
448  7316  4116           JMS     XMT12
449  7317  1156           TAD     CNTR2     /transmit block size
450  7320  7041           CIA
451  7321  4113           JMS     UPCKSM
452  7322  4116           JMS     XMT12
453  7323  5567   SD3,    JMP I   SAVE2
454
455                 /transmit the  block
456  7324  1561   SD5,    TAD I   DATPTR    /fetch word from prog. mem.
457  7325  5332           JMP     SD4
458  7326  1161   SD6,    TAD     DATPTR    /fetch data address
459  7327  6301           WRITE1  PRAM      /set it
460  7330  7200           CLA
461  7331  6300           READ1   PRAM      /fetch data word
462  7332  4113   SD4,    JMS     UPCKSM    /update checksum
463  7333  4116           JMS     XMT12     /send it off
464  7334  2161           ISZ     DATPTR    /update data pointer
465  7335  0017   K17,    17                /nop
466  7336  2156           ISZ     CNTR2     /all sent?
467  7337  5323           JMP     SD3       /no - continue sending
468  7340  4105           JMS     SDEOM     /yes - terminate this packet transmission
469  7341  5550           JMP I   MAIN2P    /packet sent,   go wait on data central
470
```

```
471 [redacted]
472                 //constants
473   7342  7776    SD5P,   SD5-SD6
474   7343  7326    SD6P,   SD6
```

```
                        PAUSE


                /      ****** subroutine INXMT ******
                /
                /set the "RTS", wait for "CTS", clear checksum reg.,
                /transmit sync. code then returns at
                /call+1 with clear acc.


7344  4406    INXMTP, JMS I  DLPTR    /service data logger - maybe?
7345  6432            SKIP3 PMOD      /carrier on?
7346  7410            SKP
7347  5345            JMP    .-2      /yes - wait
7350  6426            SRTS            /set "RTS"
7351  7327            CEN6
7352  4102            JMS    TIMER
7353  5543            JMP I TOP       /no response,   bACK to ground 0
7354  6422            CTSSF           /wait for clear-to-send ("CTS") (SP1=1)
7355  5352            JMP    .-3
7356  3162            DCA    CHKSUM   /clear checksum
7357  1144            TAD    KSYNC
7360  4110            JMS    XMTB     /transmit sync. char.
7361  5521            JMP I  INXMT


                /      ssssss subroutine XMTB ssssss
                /
                /transmit the low order 8 bits of the acc.
                /return with  acc. & link = 0 when the
                /transmitter buffer is empty.

7362  6441    XMTB,   WRITE1 PUART    /transmit char.
7363  6251            NTOUT           /display char.
7364  4102            JMS    TIMER    /update clock regs.
7365  0140    K140,   140             /nop
7366  6443            SKIP2 PUART     /wait til xt buffer
7367  5366            JMP    .-1
7370  5510            JMP I  XMTB     /then return
```

```
                        PAUSE


                /terminate communications by putting tp on hook
                /resetting "SECS" counter
                /and do as the action word would have it.
                /  bit  9 - set [RETPTR] to 200
                /  bit 10 - reset system clock reg.
                /  bit 11 - set data ram reset flag

7371  1417    TERM,   TAD I  17       /fetch # secs. in curr  min.
7372  3172            DCA    SECS     /use to resync. clock
7373  1417            TAD I  17       /fetch action word
7374  7002            BSW
7375  1417            TAD I  17
7376  7012            RTR
7377  7421            MQL             /save
7400  7501            MQA             /retrieve
7401  7510            SPA             /bit 11 set?
7402  3173            DCA    RAMF     /yes - set into ram flag
7403  7620            SNL CLA         /no - bit 10 set?
7404  5207            JMP    TE1      /no
7405  3170            DCA    TIMCTR   /yes - reset system clock reg's
7406  3171            DCA    TIMCTR+1
7407  7501    TE1,    MQA             /retrieve action word
7410  7012            RTR             /set up to check bits 8,  9
7411  7700            SMA CLA         /bit 9 set?
7412  5216            JMP    TE2      /no
7413  7203            CLA IAC BSW     /yes - set [retptr]=200
7414  7104            CLL RAL
7415  3007            DCA    RETPTR
7416  3177    TE2,    DCA    SVSTAT   /clear status hold
7417  4077            JMS    REPLY    /ACKnowledge
7420  6425    TIMOUT, WCRA PMOD       /put on hook
7421  6421            MOFF            /power down
7422  5407            JMP I  RETPTR
```

```
560                          PAUSE
561
562
563                 /     ****** subroutine XMT12 ******
564                 /
565                 /transmit one data word in two 6-bit bytes
566                 /in 00.xxx.xxx format. enter with word in acc.
567                 /
568                 /     ------ uses MQ ------
569                 /
570
571   7423  7421    XMT12P, MQL             /move word into mq
572   7424  7501            MQA             /retreive
573   7425  7002            BSW             /hi byte 1st
574   7426  0332            AND    K77      /mask hi byte
575   7427  4110            JMS    XMT8     /transmit
576   7430  7721            SWP CLA         /fetch word
577   7431  0332            AND    K77      /mask lo byte
578   7432  4110            JMS    XMT8     /send it
579   7433  5516            JMP I  XMT12    /return
580
581
582
583
584
585
586
587
588
589
590
591
592                 /     ****** subroutine TIMER ******
593                 /
594                 /whenever entered with a nonzero acc.,   routine is initialized
595                 /such that subsequent entries with a zero acc.  will cause a
596                 /time out after (ac) secs. return is at call+2 for
597                 /an initializing entry. zero acc. entries will return at call+2
598                 /unless a time-out occurs.    then a return is made to call+1.
599
600   7434  7450    TIMERP, SNA             /init. entry?
601   7435  5240            JMP    TIM1     /no
602   7436  7041            CIA             /yes - negate
603   7437  3155            DCA    CNTR1    /save as counter
604   7440  6243    TIM1,   SKIP2 PCLK      /clock tick?
605   7441  5247            JMP    TIMEXT   /no - exit to call+2
606   7442  2171            ISZ    TIMCTR+1 /yes - update system clock
607   7443  7410            SKP
608   7444  2170            ISZ    TIMCTR
609   7445  2172            ISZ    SECS     /bump sec's
610   7446  2155            ISZ    CNTR1    /bump counter
611   7447  2102    TIMEXT, ISZ    TIMER    /bump return pointer
612   7450  6230            CLEAR           /prevent autostarting
613   7451  7300            CLA CLL         /clear the garbage
614   7452  5502            JMP I  TIMER
615
```

```
                /     ****** subroutine reply ******
                /
                /return an acceptance (ACK) or rejection (NAK) of the
                /currently received message packet from data central.
                /sends "NAK" if routine is entered fwith a non-zero
                /acc. otherwise an "ACK".
                /
                /     note: if data return is expected,   data packet will be
                /       sent in lieu of an "ACK".

7453  7640      REPLYP, SZA CLA          /determine if "NAK" or "ACK"
7454  1324              TAD    KNMA
7455  1327              TAD    KACK
7456  3167              DCA    SAVE2     /  and save
7457  4121              JMS    INXMT     /init. for transmission
7460  1167              TAD    SAVE2     /fetch char.
7461  4110              JMS    XMT8      /transmit char.
7462  4110              JMS    XMT8      /flush it out
7463  4110              JMS    XMT8
7464  6427              CRTS             /terminate transmission
7465  5477              JMP I  REPLY     /return
                /     ****** subroutine gtchar ******
                /
                /waits for a character to be received,   then returns at
                /call+1 with character in acc. unless timeout occurs
                /then returns to MAIN2.
                /
                /enter with non-zero acc. if "TIMER" is to be
                /initialized.
7466  4102      GTCHRP, JMS    TIMER
7467  5550              JMP I  MAIN2P
7470  6442              SKIPI  PUART
7471  5269              JMP    .-3
7472  6440              RDARS  PUART
7473  6231              RTRUT
7474  0340              AND    K7177     /eliminate parity & extraneous bits
7475  5333              JMP I  GTCHAR
```

```
                        PAUSE

                /       ****** subroutine EOM ******
                /
                /transmit checksum,   EOM character, clear "RTS" and wait
                /for reply. if no reply within 6 secs. timeout. if "NAK"
                /is received return at call+1. if "ACK",   return at
                /call+2 with "CHKSUM" cleared.

7476  1162  SDEOMP, TAD    CHKSUM   /fetch checksum
7477  4116          JMS    XMT12    /transmit
7500  1146          TAD    KEOM     /fetch EOM char.
7501  4110          JMS    XMT8     /send it
7502  4110          JMS    XMT8     /flush it out
7503  4110          JMS    XMT8
7504  6427          CRTS            /turn off "RTS"
.7505 5505          JMP I  SDEOM
```

```
                    PAUSE


                /     ***** subroutine UPCKSM *****
                /
                /update checksum with contents of acc. return
                /with contents of acc. intact.
                /   if link= 0 ; acc. contains full word
                /   if link= 1 ; acc. contains byte
                /
                /     ------ uses MQ ------
                /
7506  7421    UCKSMP, MQL             /save contents of acc.
7507  7501            MQA
7510  7430            SZL             /check if a single or double
7511  5320            JMP     UC1     /  checksum calc.
7512  7002            BSW
7513  0332            AND     K77
7514  1162            TAD     CHKSUM
7515  3162            DCA     CHKSUM
7516  7501            MQA
7517  0332            AND     K77
7520  1162    UC1,    TAD     CHKSUM  /add acc. to checksum
7521  3162            DCA     CHKSUM  /restore updated checksum
7522  7721            CLA SWP         /retrieve original acc.
7523  5513            JMP I   UPCKSM  /and return
```

```
                        PAUSE

                /       ****** PHPC0 ******
                /
                /     phantom page 0
                /
                /page 0 entries,   to be moved upon startup.

        7524    STPGE0= .
        0077    FWAPC0= 77

                /subroutine linkages

                        EXPUNGE
        0077    *FWAPC0
                        FIXTAB

0077    0000    REPLY,  0
0100    5501            JMP I   .+1
0101    7453            REPLYP

0102    0000    TIMER,  0
0103    5504            JMP I   .+1
0104    7434            TIMERP

0105    0000    SDEOM,  0
0106    5507            JMP I   .+1
0107    7476            SDEOMP

0110    0000    XMTB,   0
0111    5512            JMP I   .+1
0112    7362            XMTBP

0113    0000    UPCKSM, 0
0114    5515            JMP I   .+1
0115    7506            UCKSMP

0116    0000    XMT12,  0
0117    5520            JMP I   .+1
0120    7423            XMT12P

0121    0000    INXMT,  0
0122    5523            JMP I   .+1
0123    7344            INXMTP

0124    0000    READ,   0
0125    5526            JMP I   .+1
0126    7720            READP

0127    0000    BKSG,   0
0130    5531            JMP I   .+1
0131    7712            BKSGP
```

```
                        PAUSE
                /constants setup to use the holes

7524  0017    KNMA,   NAK-ACK

        7527    *.+2
7527  0106    KACK,   ACK

        7532    *.+2
7532  0077    K77,    77

        7535    *.+2
7535  7371    TERMP,  TERM

        7540    *.+2
7540  7177    K7177,  7177

        7543    *.+2
7543  7402    HALT,   HLT

        7546    *.+2
7546  0010    VRSHUN, VRSION          /'X-TIM.COMM*PA' version #

        7577    *STPGE0+SIZPG0
```

```
819                             PAUSE
820
821
822
823                     /fill program area with halts
824     7577    3000    FLHLTS, DCA     0
825     7600    1373    FH1,    TAD     KHALT
826     7601    2000            ISZ     0
827     7602    3400            DCA I   0
828     7603    1000            TAD     0
829     7604    7700            SMA CLA
830     7605    5200            JMP     FH1
831
832                     /init. key reg's
833     7606    7305            CEN2
834     7607    3007            DCA     RETPTR
835     7610    3170            DCA     TIMCTR
836     7611    3171            DCA     TIMCTR+1
837     7612    7330            CEN4K
838     7613    3177            DCA     SVSTAT  /set manual restart bit
839                     /so we will know we did it
```

```
840                             PAUSE
841
842                     /transfer instructions to page-0 that will put the
843                     /system into suspension by reseting all PIEs,   then the
844                     /auto start clock is continuously reset.
845   7614  1230      SUSPND, TAD     LPLST
846   7615  3017              DCA     17
847   7616  1231              TAD     LPFWA
848   7617  3016              DCA     16
849   7620  1227              TAD     LPCNT
850   7621  3116              DCA     LPCTR
851                     /do the transfer
852   7622  1417              TAD I   17
853   7623  3416              DCA I   16
854   7624  2116              ISZ     LPCTR
855   7625  5222              JMP     .-3
856                     /and implement
857   7626  5100              JMP     BGLOOP
858
859   7627  7762      LPCNT,  BGLOOP-LPCTR
860   7630  7631      LPLST,  BGLST-1
861   7631  0077      LPFWA,  BGLOOP-1
862
863         7632      BGLST=.
864                             EXPUNGE
865         0100      *100
866                             FIXTAB
867   0100  6007      BGLOOP, CAF
868   0101  7200              CLA
869   0102  2100              ISZ     BGLOOP
870   0103  1100              TAD     BGLOOP
871   0104  0114              AND     LM777
872   0105  7640              SZA CLA
873   0106  5100              JMP     BGLOOP
874
875                     /reset auto-restart and show id
876   0107  6250              CLRAR
877   0110  7200              CLA
878   0111  1115              TAD     LPID
879   0112  6251              STOUT
880   0113  5107              JMP     .-4
881
882   0114  0777      LM777,  777
883   0115  0000      LPID,   ID
884         0116      LPCTR=.
885
886                     /reset CLP
887         7650      *BGLST+LPCTR-BGLOOP
888
889         7647      STNID=.-1
```

```
890                             PAUSE
891
892                     /send system status
893     7650    4121    SDSTAT, JMS     INXMT   /init. to transmit
894     7651    1543            TAD I   IDP     /fetch station id #
895     7652    4113            JMS     UPCKSM  /update checksum
896     7653    4116            JMS     XMT12   /send it
897     7654    1170            TAD     TIMCTR  /fetch seconds counter hi order
898     7655    4113            JMS     UPCKSM  /update checksum
899     7656    4116            JMS     XMT12   /then send it
900     7657    1171            TAD     TIMCTR+1 /same with lo order
901     7660    4113            JMS     UPCKSM
902     7661    4116            JMS     XMT12
903     7662    1177            TAD     SVSTAT  /same with system status
904     7663    4113            JMS     UPCKSM
905     7664    4116            JMS     XMT12
906     7665    1176            TAD     RAMPTR  /same with ram pointer
907     7666    4113            JMS     UPCKSM
908     7667    4116            JMS     XMT12
909     7670    4105            JMS     SDEOM   /terminate message packet
910     7671    5550            JMP I     MAIN2P
```

```
911                     PAUSE
912
913                 ////////////////////////////////////////////////////////////////////////
914                 /
915                 /
916                 /       binary loader - X-TIM version
917                 /
918                 /       binary loader adapted for use with X-TIM system PROM.
919                 /
920                 /       binary format
921                 /       1/t - nul
922                 /       init. non-0 char. (rubout=177) initiates loading
923                 /       loading terminated by 2 consecutive rubouts
924                 /
925                 /
926                 ////////////////////////////////////////////////////////////////////////
927
928
929
930                 /entry points to the binary loader
931
932   7672  1175    LDDCL,  TAD     DCLP        /fetch digital-control-list pointer
933   7673  7410            SKP
934   7674  1174    LDSRL,  TAD     SRLP        /fetch sample-rate-list pointer
935   7675  3163    LDPROG, DCA     LPTR        /initial entry to loader
936   7676  3157            DCA     CHAR1
937   7677  3160            DCA     CHAR2
938   7700  1375            TAD     IXMTRP      /this is no time to branch to data logger
939   7701  3006            DCA     DLPTR
940   7702  1374    LDPRGC, TAD     KM170       /reentry to loader
941   7703  3152            DCA     BFCTR
942   7704  5540            JMP I   BLENT
943
944
945
946
947
948
949   7705  4140            JMS     BLENT
950   7706  4124    BINLOD, JMS     READ        /ignor 0 leader
951   7707  7650            SNA CLA
952   7710  5306            JMP     .-2
953   7711  5342            JMP     BEGIN
954
955   7712  4124    BEGGP,  JMS     READ        /get byte
956   7713  1157            TAD     CHAR1       /check if 2 consecutive rubouts hav appeared
957   7714  1373            TAD     KM2RO
958   7715  7640            SZA CLA
959   7716  2127            ISZ     BEGG        /no termination yet
960   7717  5527            JMP I   BEGG        /data,   origin, or 1/t
961
962   7720  1157    READP,  TAD     CHAR1
963   7721  3160            DCA     CHAR2
964   7722  1417            TAD I   PNTR0       /fetch a byte from buffer
965   7723  0372            AND     K177
966   7724  3157            DCA     CHAR1
967   7725  2152            ISZ     BFCTR       /more in buffer?
968   7726  7410            SKP                 /yes
969   7727  4140            JMS     BLENT       /no - go get next packet
970   7730  1160            TAD     CHAR2
971   7731  5524            JMP I   READ
972
```

```
 973
 974                    /trailercode seen
 975   7732  1153   BEND,   TAD     WORD1
 976   7733  7002           BSW
 977   7734  1154           TAD     WORD2
 978   7735  7041           CIA
 979   7736  1165           TAD     LCKSUM
 980   7737  7640           SZA CLA
 981   7740  7040   BADLOD, CMA                     /bad load, xmt "NAK"
 982   7741  5303           JMP     BINLOD-1 /all done,   return to mainline
 983
 984
 985   7742  4127   BEGIN,  JMS     BECC
 986   7743  5342           JMP     .-1             /ignor leader
 987   7744  3165   CO,     DCA     LCKSUM
 988   7745  1160           TAD     CHAR2
 989   7746  3153           DCA     WORD1
 990   7747  4124           JMS     READ
 991   7750  3154           DCA     WORD2
 992   7751  4127           JMS     BECC            /look ahead
 993   7752  5332           JMP     BEND            /trailer,   end
 994   7753  1153           TAD     WORD1
 995   7754  7106           CLL RTL;RTL;RTL
       7755  7006
       7756  7006
 996   7757  1154           TAD     WORD2
 997   7760  7420           SNL
 998   7761  5367           JMP     STORE
 999   7762  3163           DCA     LPTR
1000   7763  1153   CHEX,   TAD     WORD1
1001   7764  1154           TAD     WORD2
1002   7765  1165           TAD     LCKSUM
1003   7766  5344           JMP     CO
1004
1005   7767  3563   STORE,  DCA I   LPTR
1006   7770  2163           ISZ     LPTR
1007   7771  5363           JMP     CHEX
1008
1009   7772  0177   K177,   177
1010   7773  7402   KM2RO,  -177^2
1011         0017   PNTR0=17
1012   7774  7610   KM170,  -170
1013   7775  7344   IXMTRP, INXMTP
1014         7773   KHALT=KM2RO
```

```
TUE, APR 01 1980
1015                            PAUSE
1016
1017
1018            0006    DLPTR=6
1019            0007    RETPTR= 7
1020                            EXPUNGE         /disable output
1021
1022            0152    *ENDPC0
1023    0152    0000    BFCTR,  0
1024    0153    0000    WORD1,  0
1025    0154    0000    WORD2,  0
1026    0155    0000    CNTR1,  0
1027    0156    0000    CNTR2,  0
1028    0157    0000    CHAR1,  0
1029    0160    0000    CHAR2,  0
1030    0161    0000    DATPTR, 0
1031    0162    0000    CHKSUM, 0
1032    0163    0000    LPTR,   0
1033    0164    0000    PKTSIZ, 0
1034    0165    0000    LCKSUM, 0
1035    0166    0000    SAVE1,  0
1036    0167    0000    SAVE2,  0
1037            0170    LAST0=.
1038
1039
1040
1041                    /shared page 0 locations
1042            0170    *170
1043    0170    0000    TIMCTR, 0;0
        0171    0000
1044    0172    0000    SECS,   0
1045    0173    0000    RAMF,   0               //      /pointer to data memory
1046    0174    0000    SRLP,   0               /scan rate list pointer
1047    0175    0000    DCLP,   0               /digital control list pointer
1048    0176    0000    RAMPTR, 0               /data block pointe
1049    0177    0000    SVSTAT, 0               /status reg. from previous reset
1050
1051
1052
1053
1054
1055            3600    BUFR= 3600              /buffer must be at end of ram
1056
1057                            FIXTAB          /enable output
1058
1059            7776    *7776
1060    7776    7000            START
1061    7777    5776            JMP I  .-1
1062
1063
```

PAUSE

```
1065                    PAUSE
1066
1067                ////////////////////////////////////////////////////
1068                /
1069                /     ****** X-TIM.DEF*PA ******
1070                /
1071                /     X-TIM definitions
1072                /
1073                ////////////////////////////////////////////////////
1074
1075
1076                //operating imformation
1077                /
1078                /status word format (PSTAT)
1079                /     bit 0 - manual reset occurred
1080                /         1 - auto restart occurred
1081                /         2 - modem (tp ring)
1082                /
1083                /         3 - abort call immediately
1084                /         4 - low voltage or power failure
1085                /         5 - data RAM overflow
1086                /
1087                /         6 - auto reset switch active
1088                /         7 - control switch depressed
1089                /         8 - SW8 on
1090                /
1091                /         9 - SW4 on
1092                /        10 - SW2 on
1093                /        11 - SW1 on
1094                /
1095                /adc (PADC)
1096                /     WRITE1 - set chan.# from acc. then start conversion
1097                /     READ1  - read data
1098                /     sense1 - conversion complete
1099                /
1100                /data RAM usage (PRAM)
1101                /     WRITE1 - set RAM address
1102                /     WRITE2 - store in RAM
1103                /     READ1  - read from RAM
1104                /
1105                /clock (PCLK)
1106                /     WRITE1 - reset restart counter
1107                /     sense1 - 10 hz interrupt rate
1108                /     sense2 - 1 hz interrupt rate
1109                /     sense3 - 1 min. interrupt rate
1110                /
1111                /modem (PMOD)
1112                /     WRITE1 - turn off MODEM-PROM power
1113                /     FL1 - "RTS"
1114                /     FL2 - "DTR"
1115                /     sense1 - "CTS" (+ level detect)
1116                /     sense2 - "DSR" (+ level detect)
1117                /     sense3 - "CD"  (+ level detect)
1118                /     sense4 - ring
```

```
1119                    PAUSE
1120                /
1121                /UART (PUART)
1122                /    READ1 - receiver buffer
1123                /         0 - parity error
1124                /         1 - overrun error
1125                /         2 - framing error
1126                /         5-11 - character received
1127                /    FL1 - UART reset
1128                /    sense1 - receiver ready
1129                /    sense2 - transmitter ready
1130                /    sense3 - transmitter done
1131                /    sense4 - parity error
```

```
1132                                PAUSE
1133
1134
1135        6004    RCRA=   6004
1136        6005    WCRA=   6005
1137        6015    WCRB=   6015
1138        6014    WVR=    6014
1139        6000    READ1=  6000
1140        6010    READ2=  6010
1141        6001    WRITE1= 6001
1142        6011    WRITE2= 6011
1143        6002    SKIP1=  6002
1144        6003    SKIP2=  6003
1145        6012    SKIP3=  6012
1146        6013    SKIP4=  6013
1147        6006    SFL1=   6006
1148        6016    SFL3=   6016
1149        6007    CFL1=   6007
1150        6017    CFL3=   6017
1151        0020    SP1=    20
1152        0040    SP2=    40
1153        0100    SP3=    100
1154        0200    SP4=    200
1155        0400    SL1=    400
1156        1000    SL2=    1000
1157        2000    SL3=    2000
1158        4000    SL4=    4000
1159        0400    FL1=    400
1160        1000    FL2=    1000
1161        2000    FL3=    2000
1162        4000    FL4=    4000
1163        0040    WP1=    40
1164        0200    WP2=    200
1165        0001    IE1=    1
1166        0002    IE2=    2
1167        0004    IE3=    4
1168        0010    IE4=    10
1169        7776    CIE1=   -IE1-1
1170        7775    CIE2=   -IE2-1
1171        7773    CIE3=   -IE3-1
1172        7767    CIE4=   -IE4-1
1173        0240    PSTAT=  12^20
1174        0240    PCLK=   12^20
1175        0300    PRAM=   14^20
1176        0320    PADC=   15^20
1177        0340    PDICI=  16^20
1178        0360    PDICO=  17^20
1179        0400    PTAPE=  20^20
1180        0420    PMOD=   21^20
1181        0440    PUART=  22^20
```

```
1182                        PAUSE
1183
1184
1185
1186
1187
1188                /specialized instructions
1189        6421    MOFF=   WRITE1 PMOD     /power down modem
1190        6241    STCLR=  WRITE1 PSTAT    /clear status reg. bits
1191        6240    STIN=   READ1 PSTAT     /READ status reg.
1192        6251    STOUT=  WRITE2 PSTAT    /load display reg.
1193        6436    SDTR=   SFL3 PMOD       /set "DTR"
1194        6437    CDTR=   CFL3 PMOD       /clear "DTR"
1195        6426    SRTS=   SFL1 PMOD       /set "RTS"
1196        6427    CRTS=   CFL1 PMOD       /clear "RTS"
1197        6422    CTSSF=  SKIP1 PMOD      /skip on "CTS" flag
1198        6423    DSRSF=  SKIP2 PMOD      /skip on "DSR" flag
1199        6432    CDSF=   SKIP3 PMOD      /skip on "CD" flag
1200        6243    CLKSF=  SKIP2 PCLK      /skip on 1 hz clock tick
1201        6250    CLRAR=  READ2 PCLK      /clear autorestart timer
1202        6322    ADSF=   SKIP1 PADC      /skip on conversion done
1203
1204
1205
1206        7301    CEN1=   CLL CLA IAC
1207        7305    CEN2=   CLL CLA IAC RAL
1208        7325    CEN3=   CLA IAC STL RAL
1209        7327    CEN6=   CLA STL IAC RTL
1210        7303    CEN100= CEN1 BSW
1211        7330    CEN4K=  CLA STL RAR
1212        7333    CEN6K=  CLA STL IAC RTR
1213        7360    CENM1=  STL STA
1214        7364    CENM2=  STL STA RAL
1215        7346    CENM3=  STA CLL RTL
1216        0106    ACK=    106
1217        0125    NAK=    125
1218        0103    EOM=    103
1219        0123    SYNC=   123
1220        0002    STX=    002
1221        0017    SI=     017
1222        0003    ETX=    003
1223        0001    SOH=    001
1224
```

END OF PASS 2

0 ERRORS DETECTED

SYMBOL TABLE

ACK	0106	ADSF	6322	BADLOD	7740	BADREC	7134	BDRECP	0151	BEGG	0127	BEGGP	7712	BEGIN	7742
BEND	7732	BFCTR	0152	BCLOOP	0100	BCLST	7632	BINLOD	7706	BLENT	0140	BUFR	3600	BUFRST	0147
CDSF	6432	CDTR	6437	CFL1	6007	CFL3	6017	CHAR1	0157	CHAR2	0160	CHEX	7763	CHKSUM	0162
CIE1	7776	CIE2	7775	CIE3	7773	CIE4	7767	CLKSF	6243	CLRAR	6250	CNTR1	0155	CNTR2	0156
CONT	7203	CRTS	6427	CTSSF	6422	DATPTR	0161	DCLP	0175	DESTIN	7100	DLPTR	0006	DSRSF	6423
ENDLST	7272	ENDPG0	0152	EOM	0103	ETX	0003	FH1	7600	FL1	0400	FL2	1000	FL3	2000
FL4	4000	FLHLTP	7042	FLHLTS	7577	FWAPG0	0077	GEN1	7301	GEN100	7303	GEN2	7305	GEN3	7325
GEN4K	7330	GEN6	7327	GEN6K	7333	GENM1	7360	GENM2	7364	GENM3	7346	GO	7744	GTCHAR	0135
GTCHRP	7466	HALT	7543	ID	0000	IDP	0143	IE1	0001	IE2	0002	IE3	0004	IE4	0010
INREC	0132	INRECP	7103	INXMT	0121	INXMTP	7344	IXMTRP	7775	K10	7036	K140	7365	K17	7335
K177	7772	K20	7037	K200	7040	K7	7113	K7177	7540	K7277	7041	K77	7532	K7760	7273
KACK	7527	KD30	7124	KEOM	0146	KHALT	7773	KINXMT	7075	KM170	7774	KM2R0	7773	KNMA	7524
KSL123	7101	KSPALL	7102	KSYNC	0144	LAST0	0170	LCKSUM	0165	LDDCL	7672	LDPRGC	7702	LDPROG	7675
LDSRL	7674	LM777	0114	LPCNT	7627	LPCTR	0116	LPFWA	7631	LPID	0115	LPLST	7630	LPTR	0163
MAIN2	7137	MAIN2P	0150	MN1	7157	MOFF	6421	NAK	0125	ORIGIN	7000	PADC	0320	PCLK	0240
PDIG1	0340	PDIGO	0360	PKTSIZ	0164	PMOD	0420	PNTR0	0017	PRAM	0300	PSTAT	0240	PTAPE	0400
PTRCNT	7076	PTRLST	7261	PUART	0440	RAMF	0173	RAMPTR	0176	RCRA	6004	READ	0124	READ1	6000
READ2	6010	READP	7720	REPLY	0077	REPLYP	7453	RETPTR	0007	RST	7021	SAVE1	0166	SAVE2	0167
SD1	7277	SD3	7323	SD4	7332	SD5	7324	SD5P	7342	SD6	7326	SD6P	7343	SDDATA	7275
SDEOM	0105	SDEOMP	7476	SDPROG	7274	SDSTAT	7650	SDTR	6436	SECS	0172	SFL1	6006	SFL3	6016
SI	0017	SIZPG0	0053	SKIP1	6002	SKIP2	6003	SKIP3	6012	SKIP4	6013	SL1	0400	SL2	1000
SL3	2000	SL4	4000	SOH	0001	SOURCE	7077	SP1	0020	SP2	0040	SP3	0100	SP4	0200
SRLP	0174	SRTS	6426	START	7000	STCLR	6241	STIN	6240	STNID	7647	STORE	7767	STOUT	6251
STPGE0	7524	STX	0002	SUSP	7043	SUSPND	7614	SVSTAT	0177	SYNC	0123	TE1	7407	TE2	7416
TERM	7371	TERMP	7535	TIM1	7440	TIMCTR	0170	TIMER	0102	TIMERP	7434	TIMEXT	7447	TIMOUT	7420
TOP	0145	TPR1	7052	TPRING	7044	UC1	7520	UCKSMP	7506	UPCKSM	0113	VRSHUN	7546	VRSION	0010
WCRA	6005	WCRB	6015	WORD1	0153	WORD2	0154	WP1	0040	WP2	0200	WRITE1	6001	WRITE2	6011
WVR	6014	XMT12	0116	XMT12P	7423	XMT8	0110	XMT8P	7362						

```
                /        PAUSE
                /analog to digital converter service routine
                /
                /this routine searches the channel rate list for active
                /channels (non-0) then checks if the lapse time since
                /the last sampling is equal to the rate constant. if so
                /a sample is taken, the lapse time counter is zeroed
                /and the channel # and conversion value are stored
                /in the data buffer.
                /
                /     !!!!!! note !!!!!!
                /channel 7 is optionally wired to monitor amp. offset (flag 1 =0)
                /or powersupply voltage (flag 1 = 1).

0466  3036    ADSR,   DCA     SAMPF   /clear time-to-sample flag
0467  1020            TAD     RTIME   /fetch time of sample
0470  3021            DCA     STIME   /save for updating store
0471  3373            DCA     ACHNL   /init. curr. chan. index
0472  1066            TAD     KMACHN  /init. analog chan. counter
0473  3366            DCA     CHNCTR

                /fetch rate constant and elapse time for current
                /channel, if equal then sample
0474  1373    ADSR2,  TAD     ACHNL   /fetch curr. chan.
0475  1174            TAD     SRLP    /add fwa of rate list
0476  3371            DCA     AHLD1
0477  1373            TAD     ACHNL   /fetch curr. chan.
0500  1055            TAD     DBUFP   /add fwa of data buffer
0501  3370            DCA     APNTR
0502  1771            TAD  I  AHLD1   /fetch constant
0503  7650            SNA CLA         /non 0?
0504  5326            JMP     ADSR3   /no
0505  1373            TAD     ACHNL   /yes - fetch curr. chan.
0506  1054            TAD     ETLP    /add fwa of elapse-time-list
0507  3372            DCA     AHLD2
0510  2772            ISZ  I  AHLD2   /incr. elapse time for curr. chan.
0511  1772            TAD  I  AHLD2   /fetch elapse time for curr. chan.
0512  7041            CIA             /negate
0513  1771            TAD  I  AHLD1   /add rate constant for curr. chan.
0514  7740            SMA SZA CLA     /are they =?
0515  5345            JMP     ADSR1   /no - bypass this chan.
0516  3772            DCA  I  AHLD2   /yes - zero elapse time reg.
0517  1373            TAD     ACHNL   /fetch curr. chan.
0520  1351            TAD     SFBRCH  /set up to branch to special feature
0521  3322            DCA     .+1
0522  7402            HLT             /go there

0523  1373    ADSR4,  TAD     ACHNL
0524  6321            WRITE1 PADC     /start conv.
0525  4062            JMS     ADP     /return to main loop during conversion
0526  3770    ADSR3,  DCA  I  APNTR   /back with sample value, store datum in data buffer
0527  3031            DCA     ADCVF   /clear flag
0530  2037            ISZ     UPDTF   /set update flag

                /display datum if channel switch is selected
0531  6240            STIN
0532  0367            AND     K17A
0533  7041            CIA
0534  1373            TAD     ACHNL
0535  7640            SZA CLA
0536  5345            JMP     ADSR1
0537  1770            TAD  I  APNTR
```

```
 365   0540  7004            RAL             /convert to signed octal
 366   0541  7020            CML
 367   0542  7010            RAR
 368   0543  6251            STOUT
 369   0544  7200            CLA
 370   0545  2373  ADSR1,     ISZ     ACHNL   /incr. curr. chan. index
 371   0546  2366              ISZ     CHNCTR  /more chans. to do?
 372   0547  5274              JMP     ADSR2   /yes
 373   0550  5461              MNRET
 374
 375                 /special fea tures dispatch table
 376   0551  5752  SFBRCH,   JMP I   .+1
 377   0552  0600              SFCH00
 378   0553  0601              SFCH01
 379   0554  0602              SFCH02
 380   0555  0603              SFCH03
 381   0556  0604              SFCH04
 382   0557  0617              SFCH05
 383   0560  0620              SFCH06
 384   0561  0621              SFCH07
 385   0562  0622              SFCH10
 386   0563  0623              SFCH11
 387   0564  0624              SFCH12
 388   0565  0625              SFCH13
 389
 390
 391   0566  0000  CHNCTR,  0
 392   0567  0017  K17A,    17
 393   0570  0000  APNTR,   0
 394   0571  0000  AHLD1,   0
 395   0572  0000  AHLD2,   0
 396   0573  0000  ACHNL,   0
```

```
     TUE, APR 01 1980
397                          PAUSE
398
399            0600          FIELD
400
401                    /special fea tures routines - analog channels
402
403   0600  5616     SFCH00, JMP I  ADSR4P   /chan. 0
404
405   0601  5616     SFCH01, JMP I  ADSR4P   /chan. 1
406
407   0602  5616     SFCH02, JMP I  ADSR4P   /chan. 2
408
409   0603  5616     SFCH03, JMP I  ADSR4P   /chan. 3
410
411   0604  5616     SFCH04, JMP I   ADSR4P  /chan. 4
412                    /sfch04, isz     sf4lpp /chan. 4
413                    /         jmp i  sf4lpp
414                    /sf4lpp,        sf4lp-1
415                    /special fea ture for chan. 4
416                    /sf4lp,  jmp i  adsr4p /sample normally
417                    /        jmp    s4f1   /turn on flag-1, no sample
418                    /        jmp i  adsr4p /sample normally
419                    /        jmp    c4f1   /turn off flag-1, no sample
420                    /        jmp i  adsr4p /sample normally
421                    /        jmp    s4f3   /turn on flag-3, no sample
422                    /        jmp i  adsr4p /sample normally
423                    /        jmp    c4f3   /turn off flag-3, no sample
424                    /        tad    sf4bgp /reset entry to loop
425                    /        dca    sf4lpp
426                    /        jmp    sf4lp
427
428   0605  6346     S4F1,     SFL1 PDIGI     /set flag-1
429   0606  5615               JMP I  ADSR3P
430   0607  6347     C4F1,      CFL1 PDIGI     /clear flag-1
431   0610  5615               JMP I  ADSR3P
432   0611  6356     S4F3,      SFL3 PDIGI     /set flag-3
433   0612  5615               JMP I  ADSR3P
434   0613  6357     C4F3, CFL3 PDIGI          /clear flag-3
435   0614  5615               JMP I  ADSR3P
436
437   0615  0526     ADSR3P,       ADSR3
438   0616  0523     ADSR4P,       ADSR4
439                    /sf4bgp,       sf4lp
440
441   0617  5616     SFCH05, JMP I  ADSR4P   /chan. 5
442
443   0620  5616     SFCH06, JMP I  ADSR4P   /chan. 6
```

```
                        PAUSE


                /special feature for chan. 7. normally used for monitoring
                / power supply voltage.

0621  5616    SFCH07, JMP I  ADSR4P   /sample


                /special features routines - digital channels

0622  5615    SFCH10, JMP I  ADSR3P   /chan. 10
0623  5615    SFCH11, JMP I  ADSR3P   /chan. 11
0624  5615    SFCH12, JMP I  ADSR3P   /chan. 12
0625  5615    SFCH13, JMP I  ADSR3P   /chan. 13
```

```
463                         PAUSE
464
465
466                 /adc interrupt service routine
467                 /**level 0**
468
469   0626  6320    ADCI1,  READ1 PADC      /fetch conversion value
470   0627  1240            TAD     K4000   /convert to unsigned
471   0630  7450            SNA             /can't be 0
472   0631  7001            IAC
473   0632  7040            CMA
474   0633  7450            SNA             /or 7777
475   0634  7001            IAC
476   0635  7040            CMA
477   0636  3031            DCA     ADCVF   /put into buffer
478   0637  5457            L0RET           /and return
479
480   0640  4000    K4000,  4000
```

```
      TUE. APR 01 1980
481                        PAUSE
482
483                  /*************************************************************************
484
485                  /                    clock servicing routine
486
487
488   0641  1311   CLK1,    TAD     KM1       /decrement flag count
489   0642  3032           DCA     TENTHF
490   0643  6001           ION
491   0644  2315           ISZ     TENTHS
492   0645  1315           TAD     TENTHS
493   0646  1312           TAD     KKM10
494   0647  7510           SPA
495   0650  5461           MNRET
496   0651  3315           DCA     TENTHS
497   0652  5256           JMP     .+4
498
499   0653  1311   CLK2,    TAD     KM1       /decrement flag count
500   0654  3033           DCA     SECF
501   0655  6001           ION
502   0656  2171           ISZ     CLKCTR+1 /bump clock counter, d.p. wise
503   0657  5262           JMP      .+3
504   0660  2170           ISZ     CLKCTR
505   0661  7000           NOP
506   0662  2172           ISZ     SECS      /bump secs. counter
507   0663  1172           TAD     SECS      /check for 1 minute overflow
508   0664  1314           TAD     KM60
509   0665  7510           SPA
510   0666  5461           MNRET             /retrun if not
511   0667  3172           DCA     SECS      /reset if so
512   0670  5274           JMP      .+4
513
514   0671  1311   CLK3,    TAD      KM1      /decrement flag count
515   0672  3034           DCA      MINF
516   0673  6001           ION
517   0674  2020           ISZ     RTIME     /bump RAM time
518   0675  7000           NOP
519   0676  2022           ISZ     MINS      /bump minutes
520   0677  2036           ISZ     SAMPF     /set sample time flag
521   0700  1022           TAD     MINS      /check for hour overflow
522   0701  1314           TAD     KM60
523   0702  7510           SPA
524   0703  5461           MNRET             /return if not
525   0704  3022           DCA     MINS
526   0705  2035           ISZ     BATTF     /set batttery check flag - defeat if chan. 7 used for da
527   0706  2023           ISZ     HOURS     /bump hour counter if so
528   0707  5461           MNRET
529   0710  5307           JMP     .-1
530
531   0711  7777   KM1,     -1
532   0712  7766   KKM10,   -12
533   0713  7750   KM30,    -30
534   0714  7704   KM60,    -74
535   0715  0000   TENTHS,  0
536                /*************************************************************************
```

```
     TUE, APR 01 1980
537                          PAUSE
538
539            1000          FIELD
540
541
542                          /interrupt service routine
543                          /**level 0**
544
545            1000          SERINT=.
546            0000          *0
547   0000     0000                0
548   0001     5456                L0CALL
549
550            1000          *SERINT
551   1000     3224                DCA     SAVACC
552   1001     7010                RAR                 /get the registers
553   1002     3225                DCA     SAVF        /save them
554   1003     7501                MQA                 /fetch MQ
555   1004     3223                DCA     SAVMQ       /save it
556   1005     6002                ION                 /vector to service routine
557   1006     4254                JMS     HALT        /unrecognized interupt
558
559
560
561
562   1007     6250     INTRET,    CLRAR               /clear autoreset counter
563   1010     7300                CLA CLL
564   1011     5612                JMP I   .+1         /return to telemetry if active
565   1012     1013     ROUTE,     .+1
566   1013     7300                CLL CLA
567   1014     1223                TAD     SAVMQ       /restore MQ
568   1015     7421                MQL
569   1016     1225                TAD     SAVF        /& link
570   1017     7004                RAL
571   1020     1224                TAD     SAVACC      /& accum.
572   1021     6001                ION
573   1022     5400                JMP I   0
574
575   1023     0000     SAVMQ,     0
576   1024     0000     SAVACC,    0
577   1025     0000     SAVF,      0
578   1026     1012     AROUTE,    ROUTE
579   1027     1013     IR1,       ROUTE+1
```

```
      TUE. APR 01 1980                    PAUSE
580
581
582           1034    *.+487774
583
584
585                   /interupt vector jumps
586                   /**level 0**
587     1034  5267    CLK1,   JMP     CLK11     /IE1, 10 hz
588     1035  5271            JMP     CLK12     /IE2, 1 hz
589     1036  5273            JMP     CLK13     /IE3, 1 min.
590     1037  4254            JMS     HALT
591
592     1040  4254    MOD1,   JMS     HALT
593     1041  4254            JMS     HALT
594     1042  4254            JMS     HALT
595     1043  5275            JMP     MODSER    /IE4
596
597     1044  5662    ADC1,   JMP I   ADC11P
598     1045  4254            JMS     HALT
599     1046  4254            JMS     HALT
600     1047  4254            JMS     HALT
601
602     1050  5663    DIC1,   JMP I    DIC0P
603     1051  5664            JMP I    DIC1P
604     1052  5665            JMP I    DIC2P
605     1053  5666            JMP I    DIC3P
606
607     1054  0000    HALT,   0
608     1055  7360            CENH1
609     1056  1254            TAD      HALT
610     1057  6251            STOUT
611     1060  7402            HLT
612     1061  5255            JMP      HALT+1
613     1062  0626    ADC11P, ADC11
614
615
616     1063  1200    DIC0P,  DIC0
617     1064  1206    DIC1P,  DIC1
618     1065  1215    DIC2P,  DIC2
619     1066  1224    DIC3P,  DIC3
```

```
620                         PAUSE
621
622                 /internal clock interupt routine
623                 /**level 0**
624
625   1067  2032    CLKI1,  ISZ     TENTHF
626   1070  5457            L0RET
627
628   1071  2033    CLKI2,  ISZ     SECF
629   1072  5457            L0RET
630
631   1073  2034    CLKI3,  ISZ     MINF
632   1074  5457            L0RET
633
634
```

```
635                     PAUSE
636
637                 /modem servicing routine
638                 /**level 0**
639
640   1075  1226    MODSER, TAD     AROUTE   /set up pointer into datalogger
641   1076  3006            DCA     DLPTR
642   1077  1311            TAD     RETP     /set up return pointer
643   1100  3007            DCA     RETPTR
644                 /do not service digital data logging during telemtry
645   1101  6344            RCRA PDIGI
646   1102  3307            DCA      HLDA
647   1103  1307            TAD      HLDA
648   1104  0310            AND      CMPIE
649   1105  6345            WCRA PDIGI
650   1106  7402            HLT              /wait for modem to grab control
651
652   1107  0000    HLDA,   0
653   1110  7760    CMPIE,  CIE1&CIE2&CIE3&CIE4
654
655
656
657                 /return here after tp call
658                 /**level 0**
659
660   1111  1112    RETP, .+1
661   1112  7200            CLA
662   1113  1227            TAD     IR1      /defeat exit to telemetry
663   1114  3212            DCA     ROUTE
664                 /reinit. modem PIE
665   1115  1334            TAD     VMODIP
666   1116  6434            WVR PMOD
667   1117  7200            CLA
668   1120  1044            TAD     KIE4
669   1121  6425            WCRA PMOD
670   1122  7200            CLA
671   1123  1050            TAD     KSP4
672   1124  6435            WCRB PMOD
673
674                 /check if data RAM is to be initialized
675   1125  7200            CLA
676   1126  1173            TAD     RAMF
677   1127  7640            SZA CLA
678   1130  4451            JMS I   INRAMP
679                 /enable digital interupts if active
680   1131  1307            TAD      HLDA
681   1132  6345            WCRA PDIGI
682   1133  5457            L0RET
683
684   1134  1040    VMODIP, MODI
685
686
687
688
```

```
689                             PAUSE
690
691             1200            FIELD
692
693
694                             /service UART-0
695                             /**level 0**
696     1200    7200    DIC0,   CLA
697     1201    6351            WRITE2 PDIC1    /set chan. to 0
698     1202    6340            READ1 PDIC1     /fetch datum
699     1203    4233            JMS     STFSTK
700     1204    0010    CH10,     10
701     1205    5457            L0RET
702
703
704
705
706
707                             /service UART-1
708                             /**level 0**
709     1206    7301    DIC1,   CEN1
710     1207    6351            WRITE2 PDIC1    /set chan. to 1
711     1210    7200            CLA
712     1211    6340            READ1 PDIC1     /fetch datum
713     1212    4233            JMS     STFSTK
714     1213    0011    CH11,     11
715     1214    5457            L0RET
716
717
718
719
720                             /service UART-2
721                             /**level 0**
722     1215    7305    DIC2,   CEN2
723     1216    6351            WRITE2 PDIC1    /set chan. to 2
724     1217    7200            CLA
725     1220    6340            READ1 PDIC1     /fetch datum
726     1221    4233            JMS     STFSTK
727     1222    0012    CH12,     12
728     1223    5457            L0RET
729
730
731
732
733
734                             /service UART-3
735                             /**level 0**
736     1224    7325    DIC3,   CEN3
737     1225    6351            WRITE2 PDIC1    /set chan. to 3
738     1226    7200            CLA
739     1227    6340            READ1 PDIC1     /fetch datum
740     1230    4233            JMS     STFSTK  /put into stack
741     1231    0013    CH13,     13
742     1232    5457            L0RET
743
744
```

```
                        PAUSE


                /stuff characters received into ascii stack. lo order byte contains
                /channel #, hi order byte contains character. call is "JMS STFSTK"
                /with character in acc. chan. # at call+1. return to call+2.
                /**level 0**

1233  0000  STFSTK, 0
1234  0244          AND     K77     /clean to 6 bit ascii
1235  7002          BSW             /move char. to hi byte
1236  1633          TAD I   STFSTK  /pick up chan. #
1237  3721          DCA I   ISTKP   /put into stack
1240  2321          ISZ     ISTKP   /move stack pointer
1241  2233          ISZ     STFSTK  /incr. return pointer
1242  2040          ISZ     DIGIF   /set digital input flag
1243  5633          JMP I   STFSTK

1244  0077  K77,     77
```

```
765                     PAUSE
766
767
768             /unstuffs the characters from the ascii stack packing 2/word and
769             /placing them in the data buffer for later processing by "update" for storing
770             /in the data RAM.
771
772  1245  7200 UNSTK,  CLA
773
774             /check if the stack is empty
775  1246  6002         IOF             /no surprises please
776  1247  1037         TAD     UPDTF   /check if data buffer in use
777  1250  7640         SZA CLA
778  1251  5461         MNRET           /it is - return now
779  1252  1321         TAD     ISTKP
780  1253  7041         CIA
781  1254  1320         TAD     OSTKP
782  1255  7640         SZA CLA         /if pointers are =, stack is empty
783  1256  5265         JMP     US1     /stack not empty
784  1257  1324         TAD     BGSTKP  /stack is empty, reset pointers to top of stack
785  1260  3321         DCA     ISTKP
786  1261  1324         TAD     BGSTKP
787  1262  3320         DCA     OSTKP
788  1263  3040         DCA     DIGIF   /  and clear flag
789  1264  5461         MNRET
790
791             /process contents of stack
792  1265  6001 US1,     ION
793  1266  1720         TAD I   OSTKP   /fetch word from stack
794  1267  0326         AND     K7700   /clean to char. only
795  1270  3317         DCA     DHOLD   /save
796  1271  1720         TAD I   OSTKP   /fetch word again
797  1272  0325         AND     K3      /clean to digital chan.#
798  1273  1327         TAD     TBUFFP  /add temp buffer pointer
799  1274  3323         DCA     DPTR2   /points to correct temp. buffer loc.
800  1275  1723         TAD I   DPTR2   /fetch contents of temp. buffer
801  1276  7640         SZA CLA         /anything in it?
802  1277  5303         JMP     US2     /yes
803  1300  1317         TAD     DHOLD   /no - retrieve datum
804  1301  3723         DCA I   DPTR2   /put into temp buff
805  1302  5315         JMP     US3
806
807  1303  1720 US2,     TAD I   OSTKP   /fetch data chan. #
808  1304  0244         AND     K77
809  1305  1055         TAD     DBUFP   /add pointer to data buffer
810  1306  3322         DCA     DPTR1   /use as pointer
811  1307  1317         TAD     DHOLD   /fetch current character
812  1310  7002         BSW             /move to lo byte
813  1311  1723         TAD I   DPTR2   /use to update
814  1312  3722         DCA I   DPTR1   /put into data buffer
815  1313  3723         DCA I   DPTR2   /clear temp buffer
816  1314  2037         ISZ     UPDTF   /set update flag
817  1315  2320 US3,     ISZ     OSTKP   /move stack pointer
818  1316  5461         MNRET
819
820  1317  0000 DHOLD,   0
821  1320  0000 OSTKP,   0
822  1321  0000 ISTKP,   0
823  1322  0000 DPTR1,   0
824  1323  0000 DPTR2,   0
825  1324  2343 BGSTKP,  BEGSTK
826  1325  0003 K3       3
```

```
827   1326  7700   K7700,  7700
828   1327  0024   TBUFFP, TBUFF
```

```
829                             PAUSE
830
831
832
833                     /for debugging
834     1330    7200    DSPLY,  CLA
835     1331    6240            STIN
836     1332    0346            AND     K17
837     1333    7450            SNA
838     1334    5730            JMP I   DSPLY
839     1335    1345            TAD     DSPLSP
840     1336    3347            DCA     DSHOLD
841     1337    1747            TAD I   DSHOLD
842     1340    3347            DCA     DSHOLD
843     1341    1747            TAD I   DSHOLD
844     1342    6251            STOUT
845     1343    7200            CLA
846     1344    5461            MNRET
847
848     1345    2323    DSPLSP, DSPLS-1
849     1346    0017    K17,    17
850     1347    0000    DSHOLD, 0
```

```
851                     PAUSE
852
853          1400       FIELD
854
855                     /     ****** update ******
856                     /check data buffer for data, write out to data-RAM
857                     /if so and check for full or new blocks.
858
859                     UPDATE,
860     1400  3037              DCA     UPDTF   /clear flag
861                     /set up to scan through data buffer
862     1401  3030              DCA     UCHNL   /clear curr. chan. pointer
863     1402  1067              TAD     TOTCHN
864     1403  3337              DCA     UCNTR
865     1404  1030    UD1,      TAD     UCHNL   /set up to fetch a datum
866     1405  1055              TAD     DBUFP
867     1406  3344              DCA     UDPNTR
868     1407  6240              STIN            /display data word if sw(8) is up
869     1410  0345              AND      KK17
870     1411  1347              TAD      KM10
871     1412  7100              CLL
872     1413  7650              SNA CLA
873     1414  7020              CML
874     1415  1744              TAD I   UDPNTR  /fetch a buff. entry
875     1416  7450              SNA             /real?
876     1417  5320              JMP     UD3     /no
877     1420  7430              SZL
878     1421  6251              STOUT           /display data word if link set
879     1422  3272              DCA     ARGB+1  /store datum as arg.
880     1423  7040    UD4,      CMA
881     1424  3340              DCA     UHOLD   /set twice-through flag
882     1425  1030              TAD     UCHNL   /fetch curr. chan.
883     1426  4460              RAMIO           /fetch block datum pointer
884     1427  0002    ARGA,       DDPTR         /displacement
885     1430  0000                0           /value returned here
886     1431  5325                JMP   UD5     /error return, chan. not active
887     1432  2340              ISZ     UHOLD   /normal return, check for branch
888     1433  5277              JMP     UD2
889     1434  1230              TAD     ARGA+1  /fetch datum count
890
891                     /check if current block is full, set up new block if so
892     1435  7002              BSW
893     1436  7041              CIA
894     1437  1230              TAD     ARGA+1
895     1440  7640              SZA CLA
896     1441  5244              JMP     .+3
897     1442  4741              JMS I   INBLKP
898     1443  5223              JMP     UD4
899
900                     /check if new block, insert time of sampling if so
901     1444  1230              TAD     ARGA+1
902     1445  0346              AND     K77B
903     1446  7440              SZA             /new block?
904     1447  5264              JMP     UD6     /no
905     1450  1021              TAD     STIME   /yes - set init. time
906     1451  3262              DCA     ARGE+1
907     1452  1021              TAD     STIME   /needs to be added to checksum
908     1453  3343              DCA     CKSHLD
909     1454  1021              TAD      STIME
910     1455  3350              DCA      BLKTIM
911     1456  7330              GEN4K           /set write bit
912     1457  1030              TAD     UCHNL   /fetch chan.
```

```
913   1460  4460              RAMIO
914   1461  0003  ARGE,         DETM            /displacement
915   1462  0000                0               /value
916   1463  7402                HLT             /should never get here
917
918   1464  1342  UD6,    TAD     KDDATA    /add displacement
919   1465  3271          DCA     ARGB      /modified displacement
920   1466  7330          GEN4K             /set write bit
921   1467  1030          TAD     UCHNL     /add chan. designator
922   1470  4460          RAMIO             /store datum
923   1471  0000   RGB,     0              /modified datum displacement
924   1472  0000           0              /datum goes here
925   1473  5325           JMP     UD5       /error return if chan. not active
926   1474  2230          ISZ     ARGA+1    /bump datum count
927   1475  7330          GEN4K             /set write bit
928   1476  5225          JMP     ARGA-2    /go replace updated datum pointer
929
930                 /update checksum for curr. block
931   1477  7360  UD2,    GENMI
932   1500  3340          DCA     UHOLD     /set twice-through
933   1501  1030          TAD     UCHNL     /fetch chan. designator
934   1502  4460          RAMIO             /fetch curr. checksum
935   1503  0005  ARGC,     DCKSM
936   1504  0000           0
937   1505  5325           JMP     UD5       /error return if chan. not active
938   1506  2340          ISZ     UHOLD     /check for branch
939   1507  5320          JMP     UD3
940   1510  1304          TAD     ARGC+1    /fetch curr. checksum
941   1511  1272          TAD     ARGB+1    /add curr. datum
942   1512  1343          TAD     CKSHLD    /  and any other entries
943   1513  7001          IAC               /add 1 for updated datum count
944   1514  3304          DCA     ARGC+1    /replace as call arg.
945   1515  3343          DCA     CKSHLD    /clear
946   1516  7330          GEN4K             /set write bit
947   1517  5301          JMP     ARGC-2    /go do it
948
949
950   1520  3744  UD3,    DCA I   UDPNTR    /clear buff. reg.
951   1521  2030          ISZ     UCHNL     /bump for next chan.
952   1522  2337          ISZ     UCNTR     /check if more
953   1523  5204          JMP     UD1       /loop if so
954   1524  5461          MNRET             /return if not
955
956                 /channel not active - buffer overflow - display 7777
957                 /for monitored channel.
958   1525  6240  UD5,    STIN
959   1526  0345          AND     KK17
960   1527  7041          CIA
961   1530  1030          TAD     UCHNL
962   1531  7640          SZA CLA
963   1532  5320          JMP     UD3
964   1533  7040          CMA
965   1534  6251          STOUT
966   1535  7200          CLA
967   1536  5320          JMP     UD3
968
969   1537  0000  UCNTR,  0
970   1540  0000  UHOLD,  0
971   1541  1643  INBLKP, INBLK
972   1542  0006  KDDATA, DDATA
973   1543  0000  CKSHLD, 0
974   1544  0000  UDPNTR, 0
975   1545  0017  KK17,   17
976   1546  0077  K77B,   77
```

```
977   1547  7770   KM10,   -10
978   1550  0000   BLKTIM, 0
```

```
979                             PAUSE
980
981             1600            FIELD
982
983                     /initialize data RAM
984                     /zero each cell, initialize a data block for each active channel
985
986     1600    0000    INRAM,  0
987     1601    7200            CLA
988
989                     /clear data RAM
990     1602    3240            DCA     IHOLD
991     1603    1240            TAD     IHOLD
992     1604    6301            WRITE1 PRAM
993     1605    7200            CLA
994     1606    6311            WRITE2 PRAM
995     1607    2240            ISZ     IHOLD
996     1610    5203            JMP     .-5
997     1611    3176            DCA     RAMPTR    /clear RAM pointer
998     1612    3030            DCA     UCHNL     /reset curr. chan. pointer
999     1613    3022            DCA     MINS      /reset elapsed time
1000    1614    3020            DCA     RTIME     /reset RAM time keeper
1001    1615    3023            DCA     HOURS
1002    1616    3173            DCA     RAMF      /clear RAM-init.-flag
1003
1004                    /fill RAM-pointer-list with -1
1005    1617    1067            TAD     TOTCHN
1006    1620    3241            DCA     ICNTR
1007    1621    7040            CMA
1008    1622    1053            TAD     RPLP
1009    1623    3010            DCA     10
1010    1624    7040    INR1,   CMA
1011    1625    3410            DCA I   10
1012
1013                    /  and initialize blocks for active channels
1014    1626    1174            TAD     SRLP      /fetch fwa of scan-rate-list
1015    1627    1030            TAD     UCHNL     /add curr. chan.
1016    1630    3242            DCA     IPNTR
1017    1631    1642            TAD I   IPNTR     /fetch scan-rate for curr. chan.
1018    1632    7640            SZA CLA           /active?
1019    1633    4243            JMS     INBLK     /yes - create a data block
1020    1634    2030            ISZ     UCHNL     /bump chan. pointer
1021    1635    2241            ISZ     ICNTR     /check if more chans.
1022    1636    5224            JMP     INR1      /loop if so
1023    1637    5600            JMP I   INRAM
1024    1640    0000    IHOLD,  0
1025    1641    0000    ICNTR,  0
1026    1642    0000    IPNTR,  0
1027
1028
```

```
1029                    PAUSE
1030                /initialize a block in data-RAM for the current channel ("CURCHN").
1031                /if no room left, deactivate chan. by inserting -1 into the
1032                /appropiate "RPL" location
1033    1643  0000  INBLK,0
1034
1035                /check if data RAM is full
1036    1644  1176          TAD     RAMPTR
1037    1645  7001          IAC
1038    1646  7640          SZA CLA
1039    1647  5255          JMP     .+6
1040    1650  7303          CLEN100
1041    1651  7421          MQL
1042    1652  1177          TAD     SVSTAT
1043    1653  7501          MQA
1044    1654  3177          DCA     SVSTAT
1045                /fetch data-RAM pointer and insert in "RPL"
1046    1655  1030          TAD     UCHNL
1047    1656  1053          TAD     RPLP
1048    1657  3360          DCA     IBPNTR
1049    1660  1176          TAD     RAMPTR
1050    1661  3760          DCA I   IBPNTR
1051
1052                /clear RAM I/O displacement vector
1053    1662  3347          DCA     ARGD
1054
1055                /clear checksum summer
1056    1663  3356          DCA     CKSM
1057
1058                /insert block partition (-1)
1059    1664  7040          CMA
1060    1665  3350          DCA     ARGD+1
1061    1666  4343          JMS     WRAM
1062
1063                /insert chan. # and sampling interval (0-4, chan.#; 5-11, sampling interval)
1064    1667  1030          TAD     UCHNL
1065    1670  1174          TAD     SRLP
1066    1671  3360          DCA     IBPNTR
1067    1672  1030          TAD     UCHNL
1068    1673  7002          BSW
1069    1674  7104          CLL RAL
1070    1675  1760          TAD I   IBPNTR
1071    1676  3350          DCA     ARGD+1
1072    1677  2347          ISZ     ARGD
1073    1700  4343          JMS     WRAM
1074
1075                /insert total # data points (0-5,tot.#; 6-11, # datum stored)
1076    1701  1030          TAD     UCHNL
1077    1702  1066          TAD     KMACHN
1078    1703  7700          SMA CLA
1079    1704  4341          JMS     CNDD      /digital chan. requires special treatment
1080    1705  1357          TAD     NDATA
1081    1706  3361          DCA     BHOLD
1082    1707  1361          TAD     BHOLD
1083    1710  7002          BSW
1084    1711  3350          DCA     ARGD+1
1085    1712  2347          ISZ     ARGD
1086    1713  4343          JMS     WRAM
1087
1088    1714  2347          ISZ     ARGD      /move past time entry
1089
1090                /insert "DCL" entry
```

```
1091   1715  1030          TAD     UCHNL
1092   1716  1175          TAD     DCLP
1093   1717  3360          DCA     IBPNTR
1094   1720  1760          TAD I   IBPNTR
1095   1721  3350          DCA     ARGD+1
1096   1722  2347          ISZ     ARGD
1097   1723  4343          JMS     WRAM
1098
1099                 /and now the checksum
1100   1724  1356          TAD     CKSM
1101   1725  3350          DCA     ARGD+1
1102   1726  2347          ISZ     ARGD
1103   1727  4343          JMS     WRAM
1104
1105                 /update "RAMPTR" to virgin area of data-RAM
1106                 /if wrap around occurs, set to -1 that will deactivate each
1107                 /channel as it comes up for a new block and set RAM-full bit of status word
1108   1730  7300          CLA CLL
1109   1731  1361          TAD     BHOLD
1110   1732  2347          ISZ     ARGD
1111   1733  1347          TAD     ARGD
1112   1734  1176          TAD     RAMPTR
1113   1735  7430          SZL
1114   1736  7240          CLA CMA
1115   1737  3176          DCA     RAMPTR
1116   1740  5643          JMP I   INBLK
```

```
1117                        PAUSE
1118
1119                /special data block size for digital channels
1120    1741  0000  CNDD,   0
1121    1742  5741          JMP I  CNDD
1122
1123
1124
1125
1126
1127                /special data-RAM write routine for curr. chan.
1128    1743  0000  WRAM,   0
1129    1744  7330          CEM4K
1130    1745  1030          TAD    UCHNL
1131    1746  4460          RAMIO
1132    1747  0000  ARCD,     0
1133    1750  0000            0
1134    1751  5643            JMP I    INBLK
1135    1752  1356          TAD    CKSM     /update checksum
1136    1753  1350          TAD    ARCD+1
1137    1754  3356          DCA    CKSM
1138    1755  5743          JMP I  WRAM
1139
1140    1756  0000  CKSM,   0
1141    1757  0072  NDATA,  100-6
1142    1760  0000  !BPNTR, 0
1143    1761  0000  BHOLD,  0
```

```
1144                            PAUSE
1145
1146            2000            FIELD
1147
1148                    /       ***** RWRAM *****
1149                    /fetch/deposit word to/from data-RAM
1150                    /       calling sequence:
1151                    /       call    ; enter with chan. # in acc.,
1152                    /         bit 0 set for write, bit 0 clear for read
1153                    /       call+1  ; displacement
1154                    /       call+2  ; word fetched or deposited
1155                    /       call+3  ; error return (no pointer in "rpl")
1156                    /       call+4  ; normal return, acc.=0
1157
1158    2000    0000    RWRAM,  0               /enter with chan. #
1159    2001    3235            DCA     RWHOLD  /save chan.# and direction bit
1160    2002    1235            TAD     RWHOLD  /fetch back
1161    2003    7004            RAL             /clear direction bit
1162    2004    7110            CLL RAR
1163    2005    1053            TAD     RPLP    /add pointer to rpl
1164    2006    3234            DCA     RWPNTR  /save
1165    2007    1634            TAD I   RWPNTR  /fetch pointer to block
1166    2010    7040            CMA
1167    2011    7450            SNA             /check if -1
1168    2012    5231            JMP     RW1     /if so, chan. not active, take error return
1169    2013    7040            CMA
1170    2014    1600            TAD I   RWRAM   /add displacement
1171    2015    6301            WRITE1 PRAM     /set address
1172    2016    7200            CLA
1173    2017    2200            ISZ     RWRAM   / move pointer to datum argument
1174    2020    1235            TAD     RWHOLD  /fetch direction bit
1175    2021    7710            SPA CLA         /set?
1176    2022    5226            JMP     RWWR    /yes - do a write-out
1177    2023    6300            READ1 PRAM      /no - do a readout
1178    2024    3600            DCA I   RWRAM   /put into argument list
1179    2025    5231            JMP     RW1     /go set up for normal return
1180    2026    1600    RWWR,   TAD I   RWRAM   /fetch word from arg. list
1181    2027    6311            WRITE2 PRAM     /deposit in RAM
1182    2030    7200            CLA
1183    2031    2200    RW1,    ISZ     RWRAM   /bump for return
1184    2032    2200            ISZ     RWRAM
1185    2033    5600            JMP I   RWRAM
1186
1187    2034    0000    RWPNTR, 0
1188    2035    0000    RWHOLD, 0
1189
1190
1191                    /display selected channel if 'CTR' is pressed
1192                    /or elapsed time if not and monitor switch is in time
1193                    /position on each pass.
1194                    /if C(SW)=17, display station id. of loaded program.
1195
1196    2036    7200    MNITOR, CLA
1197    2037    6240            MTIR
1198    2040    0302            AND     K20
1199    2041    7650            SNA CLA
1200    2042    5245            JMP     .+3
1201    2043    6240            MTIR            /if C(SW)=17, display id
1202    2044    0277            AND     [illegible]
1203    2045    1300            TAD     [illegible]
```

```
1204  2046  [illegible]
1205  2047  [illegible]  [illegible]P  [illegible]T3
1206  2050  1301          TAD     KID
1207  2051  5261          JMP     MNT1
1208  2052  6240  MNT3,   STIN
1209  2053  0276          AND     KK7
1210  2054  6321          WRITE1  PADC
1211  2055  4062          JMS      ADP      /return to main loop during conversion
1212  2056  7004          RAL               /back with sample value, convert to signed octal
1213  2057  7020          CML
1214  2060  7010          RAR
1215  2061  6251  MNT1,   STOUT
1216  2062  7200          CLA
1217  2063  3031          DCA      ADCVF    /clear flag
1218  2064  5461          MNRET
1219  2065  6240  MNT2,   STIN
1220  2066  0303          AND     K100
1221  2067  7640          SZA CLA
1222  2070  5461          MNRET
1223  2071  1023          TAD     HOURS
1224  2072  0304          AND     K77A
1225  2073  7002          BSW
1226  2074  1022          TAD     MINS
1227  2075  5261          JMP     MNT1
1228
1229  2076  0007  KK7,    7
1230  2077  0017  KKK17,  17
1231  2100  7761  KM17,   -17
1232  2101  0000  KID,    ID
1233  2102  0020  K20,    20
1234  2103  0100  K100,   100
1235  2104  0077  K77A,   77
```

```
1236                        PAUSE
1237
1238
1239
1240                /check "SRL" and "ETL" for a logging coming up within 3 minutes.
1241                /if so set abort-call bit of "SVSTAT".
1242                /     !!!!!! currently not in use !!!!!!
1243
1244  2105  0000    LKAHD,  0
1245  2106  7200            CLA
1246  2107  1067            TAD     TOTCHN  /set up for chan. count
1247  2110  3340            DCA     LCTR
1248  2111  7040            CMA             /init. pointers to lists
1249  2112  1174            TAD     SRLP
1250  2113  3010            DCA     10
1251  2114  7040            CMA
1252  2115  1054            TAD     ETLP
1253  2116  3011            DCA     11
1254
1255                /search lists
1256  2117  1410    LH1,    TAD I   10
1257  2120  7450            SNA             /check if chan. is active
1258  2121  5335            JMP     LH3     /bypass if so
1259  2122  7041            CIA
1260  2123  1337            TAD     KK3
1261  2124  1411            TAD I   11
1262  2125  7710            SPA CLA         /within 3 minutes?
1263  2126  5331            JMP     LH2     /no
1264  2127  4341            JMS     SCAB    /yes - set the "SVSTAT" bit
1265  2130  5705            JMP I   LKAHD   /  and return now
1266  2131  2340    LH2,    ISZ     LCTR      /check for more
1267  2132  5317            JMP     LH1     /loop if so
1268  2133  4351            JMS     CCAB    /if not, clear "SVSTAT" bit
1269  2134  5705            JMP I   LKAHD   /  and return
1270  2135  2011    LH3,    ISZ     11
1271  2136  5331            JMP     LH2
1272
1273  2137  0003    KK3,    3
1274  2140  0000    LCTR,   0
```

```
1275                        PAUSE
1276
1277                /set call-abort-bit of "SVSTAT"
1278                /     !!!!!! currently not in use !!!!!!
1279
1280    2141  0000  SCAB,   0
1281    2142  7303          GEN100
1282    2143  7006          RTL
1283    2144  7421          MQL
1284    2145  1177          TAD    SVSTAT
1285    2146  7501          MQA
1286    2147  3177          DCA    SVSTAT
1287    2150  5741          JMP I  SCAB
1288
1289
1290
1291
1292
1293                /clear call-abort bit of "SVSTAT"
1294                /     !!!!!! currently not in use !!!!!!
1295
1296    2151  0000  CCAB,   0
1297    2152  7303          GEN100
1298    2153  7006          RTL
1299    2154  7040          CMA
1300    2155  0177          AND    SVSTAT
1301    2156  3177          DCA    SVSTAT
1302    2157  5751          JMP I  CCAB
```

```
1303                        PAUSE
1304
1305
1306          2200          FIELD
1307
1308                    /check battery voltage (chan. 7 with FLAG-1 hi). if < 11.5 volts
1309                    /(<'6176) set low-voltage bit of "SVSTAT".
1310
1311    2200  3035    CKBATT, DCA     BATTF       /clear flag
1312    2201  1225            TAD     K7A         /select chan. 7
1313    2202  6321            WRITE1 PADC         /initiate conversion
1314    2203  4062            JMS      ADP        /return to mainloop during conversion
1315    2204  7421            MQL                 /back with sample value, save it
1316    2205  3031            DCA      ADCVF      /clear flag
1317    2206  7501            MQA                 /retrieve value
1318    2207  6327            CFL1 PADC           /conversion done, turn off relay
1319    2210  7120            STL
1320    2211  1223            TAD     MTHRSH      /check if low
1321    2212  7620            SNL CLA
1322    2213  5461            MNRET               /ok - return
1323    2214  7303            GEN100              /low - set the bit
1324    2215  7104            CLL RAL
1325    2216  7421            MQL
1326    2217  1177            TAD     SVSTAT
1327    2220  7501            MQA
1328    2221  3177            DCA     SVSTAT
1329    2222  5461            MNRET
1330
1331    2223  1602    MTHRSH, -6176
1332    2224  0000    CBCTR,  0
1333    2225  0007    K7A,    7
```

```
TUE, APR 01 1980
1334                            PAUSE
1335
1336    2226    0000            ID              /station identifier
1337    2227    0006            VRSION          /version
1338
1339
1340            2230    *.+787770
1341
1342                    /lists
1343            2230    BEGLST=.
1344                    /Scan-Rate-List (0 indicates inactive chan.)
1345    2230    0000    SRL,    0;0;0;0;0;0;0;0;0;0;0;0
        2231    0000
        2232    0000
        2233    0000
        2234    0000
        2235    0000
        2236    0000
        2237    0000
        2240    0000
        2241    0000
        2242    0000
        2243    0000
1346
1347                    /Digital-Control-List
1348    2244    0000    DCL,    0;0;0;0;0;0;0;0;0;0;0;0
        2245    0000
        2246    0000
        2247    0000
        2250    0000
        2251    0000
        2252    0000
        2253    0000
        2254    0000
        2255    0000
        2256    0000
        2257    0000
1349            2257      ENDLST=.-1
```

```
      TUE, APR 01 1980
1379                        PAUSE
1380
1381                  /page   0 storage assignments
1382
1383
1384                  /-------------------------------------------------------------------
1385
1386            0006  DLPTR= 6
1387            0007  *7
1388  0007      0200  RETPTR, START
1389
1390            0020  *20
1391            0020  STREG=.
1392  0020      0000  RTIME,  0               /data RAM time keeper
1393  0021      0000  STIME,  0               /relative time of last sample cycle
1394  0022      0000  MINS,   0               /minutes reg.
1395  0023      0000  HOURS,  0
1396
1397  0024      0000  TBUFF,  0
1398  0025      0000          0
1399  0026      0000          0
1400  0027      0000          0
1401
1402  0030      0000  UCHNL,  0               /channel currently being updated
1403  0031      0000  ADCVF,  0               /adc conversion done flag
1404  0032      0000  TENTHF, 0               /comes up each tenth sec. if activated
1405  0033      0000  SECF,   0               /comes up for each sec. tick
1406  0034      0000  MINF,   0               /comes up for each minute tick if activated
1407  0035      0000  BATTF,  0               /comes up once per hour
1408  0036      0000  SAMPF,  0               /comes up at the sample time
1409  0037      0000  UPDTF,  0               /set if sample data in buffer
1410  0040      0000  DIGIF,  0               /set if digital input occurred
1411            0041  ENDREC=       .
1412
1413                  /-------------------------------------------------------------------
1414                  /
1415                  /***constants
1416  0041      0001  KIE1,    IE1
1417  0042      0002  KIE2,    IE2
1418  0043      0004  KIE3,    IE3
1419  0044      0010  KIE4,    IE4
1420  0045      0020  KSP1,    SP1
1421  0046      0040  KSP2,    SP2
1422  0047      0100  KSP3,    SP3
1423  0050      0200  KSP4,    SP4
```

```
1424                        PAUSE
1425
1426    0051  1600    INRAMP, INRAM
1427    0052  5521    KMENDL, -ENDLST
1428    0053  2260    RPLP, RPL               /pointer to RAM-pointer-list
1429    0054  2274    ETLP, ETL               /pointer to elapsed-time-list
1430    0055  2310    DBUFP,DBUFR             /pointer to data buffer
1431          7371    TERM=   7371            /points to telemetry termination sequence
1432
1433                        DECIMAL
1434          0010    NACHN=8                 /number of analog channels
1435          0004    NDCHN=4                 /number of digital channels
1436                        OCTAL
1437          5436    L0CALL= JMP I .         /interrupt service entry
1438    0056  1000            SERINT
1439          5457    L0RET=JMP I .           /interrupt service return
1440    0057  1007            INTRET
1441          4460    RAMIO=JMS I .           /data RAM utility service routine
1442    0060  2000            RWRAM
1443          5461    MNRET=  JMP I .         /return to main loop
1444    0061  0407            MAINLP
1445
1446    0062  0000    ADP,    0               /a/d busy rturn to main loop
1447    0063  7240            CLA CMA
1448    0064  3031            DCA     ADCVF   /set a/d busy flag
1449    0065  5461            MNRET
1450
1451
1452    0066  7770    KMACHN, -NACHN
1453    0067  7764    TOTCHN, -NACHN-NDCHN    /total # of data channels
1454
1455                  /*********************************************************
1456                  /locations 77-167 used by PROM
1457                  /*********************************************************
1458
1459                  /shared locations
1460          0170    *170
1461    0170  0000    CLKCTR, 0;0
        0171  0000
1462    0172  0000    SECS,   0               /sec. count
1463    0173  0000    RAMF,   0
1464    0174  2230    SRLP,   SRL             /pointer to scan-rate-list
1465    0175  2244    DCLP,   DCL             /pointer to Digital-Control-List
1466    0176  0000    RAMPTR, 0               /points to next available location in data RAM
1467    0177  0000    SVSTAT, 0               /TIM status word
1468
1469          4000    *4000
1470    4000  0000            0               /allow for assembler residual
```

```
1471                    PAUSE
1472
1473                /************************************************************
1474                /
1475                /data-RAM block format
1476                /initialized to zeroes
1477                /block size = 64(10) for analog channels
1478                /
1479                /       0 : -1, block partition
1480                /       1 : 0-4, chan.#; 5-11, sampling interval (255(100) max.)
1481                /       2 : 0-5, total # datum; 6-11, # datum stored
1482                /       3 : 0-11, total elapsed time - minutes for 1st data entry
1483                /       4 : 0-11, digital-control-word or exponent
1484                /       5 : 0-11, checksum
1485                /       6 : data(1)
1486                /       :
1487                /       :
1488                /      77 : data(58)
1489                /
1490                /displacement defitions
1491          0000              DPART= 0
1492          0001              DCHSI= 1
1493          0002              DDPTR= 2
1494          0003              DETM=  3
1495          0004              DDCW=  4
1496          0005              DCKSM= 5
1497          0006              DDATA= 6
1498
```

```
1499                    PAUSE
1500
1501                ////////////////////////////////////////////////////
1502                /
1503                /     ****** X-TIM.DEF*PA ******
1504                /
1505                /     X-TIM definitions
1506                /
1507                ////////////////////////////////////////////////////
1508
1509
1510                //operating information
1511                /
1512                /status word format (PSTAT)
1513                /     bit 0 - manual reset occurred
1514                /         1 - auto restart occurred
1515                /         2 - modem (tp ring)
1516                /
1517                /         3 - abort call immediately
1518                /         4 - low voltage or power failure
1519                /         5 - data RAM overflow
1520                /
1521                /         6 - auto reset switch active
1522                /         7 - control switch depressed
1523                /         8 - SW8 on
1524                /
1525                /         9 - SW4 on
1526                /        10 - SW2 on
1527                /        11 - SW1 on
1528                /
1529                /adc (PADC)
1530                /     WRITE1 - set chan.# from acc. then start conversion
1531                /     READ1  - read data
1532                /     sense1 - conversion complete
1533                /
1534                /data RAM usage (PRAM)
1535                /     WRITE1 - set RAM address
1536                /     WRITE2 - store in RAM
1537                /     READ1  - read from RAM
1538                /
1539                /clock (PCLK)
1540                /     WRITE1 - reset restart counter
1541                /     sense1 - 10 hz interrupt rate
1542                /     sense2 - 1 hz interrupt rate
1543                /     sense3 - 1 min. interrupt rate
1544                /
1545                /modem (PMOD)
1546                /     WRITE1 - turn off MODEM-PROM power
1547                /     FL1 - "RTS"
1548                /     FL2 - "DTR"
1549                /     sense1 - "CTS" (+ level detect)
1550                /     sense2 - "DSR" (+ level detect)
1551                /     sense3 - "CD"  (+ level detect)
1552                /     sense4 - ring
```

```
1553                    PAUSE
1554                /
1555                /UART (PUART)
1556                /    READ1 - receiver buffer
1557                /        0 - parity error
1558                /        1 - overrun error
1559                /        2 - framing error
1560                /        5-11 - character received
1561                /    FL1 - UART reset
1562                /    sense1 - receiver ready
1563                /    sense2 - transmitter ready
1564                /    sense3 - transmitter done
1565                /    sense4 - parity error
```

TUE, APR 01 1980

```
1566                            PAUSE
1567
1568
1569        6004    RCRA=    6004
1570        6005    WCRA=    6005
1571        6015    WCRB=    6015
1572        6014    WVR=     6014
1573        6000    READ1=   6000
1574        6010    READ2=   6010
1575        6001    WRITE1=  6001
1576        6011    WRITE2=  6011
1577        6002    SKIP1=   6002
1578        6003    SKIP2=   6003
1579        6012    SKIP3=   6012
1580        6013    SKIP4=   6013
1581        6006    SFL1=    6006
1582        6016    SFL3=    6016
1583        6007    CFL1=    6007
1584        6017    CFL3=    6017
1585        0020    SP1=     20
1586        0040    SP2=     40
1587        0100    SP3=     100
1588        0200    SP4=     200
1589        0400    SL1=     400
1590        1000    SL2=     1000
1591        2000    SL3=     2000
1592        4000    SL4=     4000
1593        0400    FL1=     400
1594        1000    FL2=     1000
1595        2000    FL3=     2000
1596        4000    FL4=     4000
1597        0040    WP1=     40
1598        0200    WP2=     200
1599        0001    IE1=     1
1600        0002    IE2=     2
1601        0004    IE3=     4
1602        0010    IE4=     10
1603        7776    CIE1=    -IE1-1
1604        7775    CIE2=    -IE2-1
1605        7773    CIE3=    -IE3-1
1606        7767    CIE4=    -IE4-1
1607        0240    PSTAT=   12^20
1608        0240    PCLK=    12^20
1609        0300    PRAM=    14^20
1610        0320    PADC=    15^20
1611        0340    PDIGI=   16^20
1612        0360    PDIGO=   17^20
1613        0400    PTAPE=   20^20
1614        0420    PMOD=    21^20
1615        0440    PUART=   22^20
```

```
1616                        PAUSE
1617
1618
1619
1620
1621
1622                /specialized instructions
1623        6421    MOFF=   WRITE1 PMOD     /power down modem
1624        6241    STCLR=  WRITE1 PSTAT    /clear status reg. bits
1625        6240    STIN=   READ1 PSTAT     /READ status reg.
1626        6251    STOUT=  WRITE2 PSTAT    /load display reg.
1627        6436    SDTR=   SFL3 PMOD       /set "DTR"
1628        6437    CDTR=   CFL3 PMOD       /clear "DTR"
1629        6426    SRTS=   SFL1 PMOD       /set "RTS"
1630        6427    CRTS=   CFL1 PMOD       /clear "RTS"
1631        6422    CTSSF=  SKIP1 PMOD      /skip on "CTS" flag
1632        6423    DSRSF=  SKIP2 PMOD      /skip on "DSR" flag
1633        6432    CDSF=   SKIP3 PMOD      /skip on "CD" flag
1634        6243    CLKSF=  SKIP2 PCLK      /skip on 1 hz clock tick
1635        6250    CLRAR=  READ2 PCLK      /clear autorestart timer
1636        6322    ADSF=   SKIP1 PADC      /skip on conversion done
1637
1638
1639
1640        7301    GEN1=   CLL CLA IAC
1641        7305    GEN2=   CLL CLA IAC RAL
1642        7325    GEN3=   CLA IAC STL RAL
1643        7327    GEN6=   CLA STL IAC RTL
1644        7303    GEN100= GEN1 BSW
1645        7330    GEN4K=  CLA STL RAR
1646        7333    GEN6K=  CLA STL IAC RTR
1647        7360    GENM1=  STL STA
1648        7364    GENM2=  STL STA RAL
1649        7346    GENM3=  STA CLL RTL
1650        0106    ACK=    106
1651        0125    NAK=    125
1652        0103    EOM=    103
1653        0123    SYNC=   123
1654        0002    STX=    002
1655        0017    SI=     017
1656        0003    ETX=    003
1657        0001    SOH=    001
1658
```

END OF PASS 2

0 ERRORS DETECTED

TUE, APR 01 1980

SYMBOL TABLE

ACHNL	0573	ACK	0106	ADCI	1044	ADCII	0626	ADCIIP	1062	ADCVF	0031	ADP	0062	ADSF	6322
ADSR	0466	ADSR1	0545	ADSR2	0474	ADSR3	0526	ADSR3P	0615	ADSR4	0523	ADSR4P	0616	ADSRP	0454
AHLD1	0571	AHLD2	0572	APNTR	0570	ARGA	1427	ARGB	1471	ARGC	1503	ARGD	1747	ARGE	1461
AROUTE	1026	BATTF	0035	BEGLST	2230	BEGSTK	2343	BGSTKP	1324	BHOLD	1761	BLKTIM	1550	C4F1	0607
C4F3	0613	CBCTR	2224	CCAB	2151	CDSF	6432	CDTR	6437	CFL1	6007	CFL3	6017	CH10	1204
CH11	1213	CH12	1222	CH13	1231	CHNCTR	0566	C1E1	7776	C1E2	7775	C1E3	7773	C1E4	7767
CKBATP	0455	CKBATT	2200	CKSHLD	1543	CKSM	1756	CLK1	0641	CLK1P	0450	CLK2	0653	CLK2P	0451
CLK3	0671	CLK3P	0452	CLKCTR	0170	CLK1	1634	CLK11	1067	CLK12	1071	CLK13	1073	CLKSF	6243
CLRAR	6250	CMP1E	1110	CRTS	6427	CTSSF	6422	DBUFP	0055	DBUFR	2310	DCHS1	0001	DCKSM	0005
DCL	2244	DCLP	0175	DDATA	0006	DDCW	0004	DDPTR	0002	DETM	0003	DHOLD	1317	DIC0	1200
DIC0P	1063	DIC1	1206	DIC1P	1064	DIC2	1215	DIC2P	1065	DIC3	1224	DIC3P	1066	DIC1	1050
DIC1F	0040	DLPTR	0006	DPART	0000	DPTR1	1322	DPTR2	1323	DSHOLD	1347	DSPLS	2324	DSPLSP	1345
DSPLY	1330	DSPLYP	0465	DSRSF	6423	ENDLST	2257	ENDREG	0041	EOM	0103	ETL	2274	ETLP	0054
ETX	0003	FL1	0400	FL2	1000	FL3	2000	FL4	4000	GEN1	7301	GEN100	7303	GEN2	7305
GEN3	7325	GEN4K	7330	GEN6	7327	GEN6K	7333	GENM1	7360	GENM2	7364	GENM3	7346	GNDD	1741
C00	0217	C01	0230	C01A	0235	C02	0261	C03	0353	HALT	1054	HLDA	1107	HOURS	0023
IBPNTR	1760	ICNTR	1641	ID	0000	IE1	0001	IE2	0002	IE3	0004	IE4	0010	IHOLD	1640
INBLK	1643	INBLKP	1541	INR1	1624	INRAM	1600	INRAMP	0051	INT11	0333	INT12	0341	INT13	0347
INTRET	1007	IPNTR	1642	IR1	1027	ISTKP	1321	ISTKPP	0456	K100	2103	K17	1346	K17A	0567
K20	2102	K3	1325	K4000	0640	K7	0366	K77	1244	K7700	1326	K777	0370	K77A	2104
K77B	1546	K7A	2225	KDDATA	1542	KID	2101	KIE1	0041	KIE2	0042	KIE3	0043	KIE4	0044
KIEN	0375	KK17	1545	KK3	2137	KK7	2076	KKK17	2077	KKM10	0712	KM1	0711	KM10	1547
KM14	0367	KM17	2100	KM30	0713	KN60	0714	KNACHN	0066	KNENDL	0052	KSP1	0045	KSP2	0046
KSP3	0047	KSP4	0050	LOCALL	5456	LORET	5457	LCTR	2140	LH1	2117	LH2	2131	LH3	2135
LKAHD	2105	MAIN1	0436	MAINLP	0407	MINF	0034	MINS	0022	MNITOR	2036	MNITRP	0453	MNRET	5461
MNT1	2061	MNT2	2065	MNT3	2052	MOD1	1040	MODSER	1075	MOFF	6421	MTHRSH	2223	NACHN	0010
NAK	0125	NDATA	1757	NDCHN	0004	OSTKP	1320	OSTKPP	0457	PADC	0320	PBCSTK	0460	PCLK	0240
PDIC1	0340	PDIC0	0360	PMOD	0420	PRAM	0300	PSTAT	0240	PTAPE	0400	PUART	0440	RAMF	0173
RAM10	4460	RAMPTR	0176	RCNTR	0376	RCRA	6004	READ1	6000	READ2	6010	RECCNT	0365	RETP	1111
RETPTR	0007	ROUTE	1012	ROUTE1	0463	ROUTEP	0462	RPL	2260	RPLP	0053	RTIME	0020	RW1	2031
RWHOLD	2035	RWPNTR	2034	RWRAM	2000	RWWR	2026	S4F1	0605	S4F3	0611	SAMPF	0036	SAVACC	1024
SAVF	1025	SAVMQ	1023	SCAB	2141	SDTR	5436	SECF	0033	SECS	0172	SERINT	1000	SETMIN	0213
SFBRCH	0551	SFCH00	0600	SFCH01	0601	SFCH02	0602	SFCH03	0603	SFCH04	0604	SFCH05	0617	SFCH06	0620
SFCH07	0621	SFCH10	0622	SFCH11	0623	SFCH12	0624	SFCH13	0625	SFL1	6006	SFL3	6016	SI	0017
SKIP1	6002	SKIP2	6003	SKIP3	6012	SKIP4	6013	SL1	0400	SL2	1000	SL3	2000	SL4	4000
SOH	0001	SP1	0020	SP2	0040	SP3	0100	SP4	0200	SRL	2230	SRLP	0174	SRTS	6426
START	0200	STCLR	6241	STFSTK	1233	STIME	0021	STIN	6240	STOUT	6251	STREG	0020	STREGP	0364
STX	0002	SVSTAT	0177	SYNC	0123	TBUFF	0024	TBUFFP	1327	TENTHF	0032	TENTHS	0715	TERM	7371
TOTCHN	0067	UCHNL	0030	UCNTR	1537	UD1	1404	UD2	1477	UD3	1520	UD4	1423	UD5	1525
UD6	1464	UDPNTR	1544	UHOLD	1540	UNSTK	1245	UNSTKP	0461	UPDATE	1400	UPDATP	0464	UPDTF	0037
US1	1265	US2	1303	US3	1315	VADC1	0371	VCLK1	0373	VDIC1	0374	VMOD1	0372	VMOD1P	1134
VRSION	0006	WCRA	6005	WCRB	6015	WP1	0040	WP2	0200	WRAM	1743	WRITE1	6001	WRITE2	6011
WVR	6014														